Lattice-Based Succinct Arguments for NP with Polylogarithmic-Time Verification [*]

Jonathan Bootle
jbt@zurich.ibm.com
IBM Research – Zurich

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Katerina Sotiraki
katesot@berkeley.edu
UC Berkeley

June 16, 2023

Abstract

Succinct arguments that rely on the Merkle-tree paradigm introduced by Kilian (STOC 92) suffer from larger proof sizes in practice due to the use of generic cryptographic primitives. In contrast, succinct arguments with the smallest proof sizes in practice exploit homomorphic commitments. However these latter are quantum insecure, unlike succinct arguments based on the Merkle-tree paradigm.

A recent line of works seeks to address this limitation, by constructing quantum-safe succinct arguments that exploit lattice-based commitments. The eventual goal is smaller proof sizes than those achieved via the Merkle-tree paradigm. Alas, known constructions lack succinct verification.

In this paper, we construct the first interactive argument system for NP with succinct verification that, departing from the Merkle-tree paradigm, exploits the homomorphic properties of lattice-based commitments. For an arithmetic circuit with N gates, our construction achieves verification time $\mathsf{polylog}(N)$ based on the hardness of the Ring Short-Integer-Solution (RSIS) problem.

The core technique in our construction is a delegation protocol built from commitment schemes based on leveled bilinear modules, a new notion that we deem of independent interest. We show that leveled bilinear modules can be realized from pre-quantum and from post-quantum cryptographic assumptions.

Keywords: succinct arguments; lattices; short-integer-solution problem

[*]This is the full version of the CRYPTO'23 paper.

Contents

1 Introduction

Succinct arguments enable an untrusted prover to convince a skeptical verifier that a given computation is correctly executed, while incurring communication complexity, and sometimes also verification time, that is much smaller than the computation size. Succinct arguments were first constructed by Kilian in [Kil92], and since then much research has been devoted to improving their efficiency and security. Kilian shows how to compile a PCP into a succinct argument by using a Merkle tree, given any collision-resistant hash function. This "Merkle-tree paradigm" can also be used to construct succinct arguments from IOPs [BCS16; RRR21], which are more efficient generalizations of PCPs (and, in particular, are used in practice).

In anticipation of the threat of quantum computers, cryptographers have started investigating quantum-safe constructions of succinct arguments. Kilian's construction is such a construction: recent work [CMSZ21] establishes that Kilian's interactive argument is quantum-safe if the used hash function is quantum-safe.

Split-and-fold techniques in the pre-quantum setting: a success story. Departing from the Merkle-tree paradigm, an approach based on *split-and-fold techniques* [BCCGP16; BBBPWM18; LMR19; BFS20; Lee21] has led to succinct arguments that are remarkably efficient and successful in practice. Even though asymptotically these constructions have similar proof sizes to constructions based on Merkle trees, in practice, they obtain smaller proofs by exploiting the algebraic structure of homomorphic commitment schemes.

This approach has several advantages over Merkle-tree constructions beyond smaller communication complexity. For example, the sumcheck protocol [LFKN92] underlies split-and-fold techniques [BCS21], which facilitates space-efficient constructions [BHRRS20; BHRRS21]. In contrast, no space-efficient constructions are known for succinct arguments based on Merkle trees.

Unfortunately, the required homomorphic commitment schemes are known *only from pre-quantum cryptography that relies on groups and bilinear groups*.

What happens in the post-quantum setting? The success story of split-and-fold techniques in the pre-quantum setting has motivated a line of work studying similar approaches in the post-quantum setting using lattices [BLNS20; BCS21; ACK21; AL21]. The eventual goal is to achieve succinct arguments from lattice-based split-and-fold techniques that have better efficiency compared to their Merkle-tree-based counterparts (and possibly have other benefits such as space efficiency). In the meantime, the cited works have laid initial foundations for such succinct arguments, but more work is needed to achieve this goal.

The inspiration comes from quantum-safe constructions of signature schemes, where using the algebraic structure of lattices eventually led to shorter signatures compared to using hash functions. For instance, among the standardization candidates in the NIST Post-Quantum Competition [NIS16], lattice-based signature schemes such as Falcon [Fal] and Dilithium [Dil] offer shorter signatures compared to hash-based signatures such as SPHINCS+ [Sph] and Picnic [Pic].

Succinct verification. The above lattice-based succinct arguments lack succinct verification (the time complexity of the verifier is at least the time of the proved computation). This is in contrast to constructions based on Merkle trees (and some pre-quantum constructions based on split-and-fold techniques [BFS20; Lee21]), which offer succinct verification. This leads to the main question motivating our work:

How to construct interactive arguments with succinct verification
from split-and-fold techniques based on lattices?

1.1 Our results

We answer this question in the affirmative, achieving succinct verification for R1CS, a popular circuit-like NP problem, in the *preprocessing setting*.

Definition 1 (informal)**.** *The R1CS problem over a ring $R_\bullet$ asks: given coefficient matrices $A, B, C \in R_\bullet^{N \times N}$ each containing at most $M = \Omega(N)$ non-zero entries, and an instance vector $\underline{x}$ over $R_\bullet$, is there a witness vector $\underline{w}$ over $R_\bullet$ such that $\underline{z} := (\underline{x}, \underline{w}) \in R_\bullet^N$ and $A\underline{z} \circ B\underline{z} = C\underline{z}$? (Here "$\circ$" is the entry-wise product.)*

In the preprocessing setting, an *indexer* algorithm performs a public computation that depends on the coefficient matrices A, B and C (the "circuit description"), leading to a long proving key and a short verification key. Thereafter, anyone can use the proving/verification key to prove/verify statements for the preprocessed coefficient matrices. The argument verifier may achieve succinct verification because it only needs the verification key and the instance vector $\underline{x}$, with no need to read the (much larger) coefficient matrices. (Non-uniform computations require some form of preprocessing to enable succinct verification.)

We construct a succinct interactive argument with preprocessing for the R1CS problem over rings.

Theorem 1 (informal)**.** *Let $R := \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$ where Φ_d is the d-th cyclotomic polynomial and d is a prime power. Let p, q be primes such that $p \ll q$. If the SIS problem is hard over R/qR then there is a preprocessing interactive argument of knowledge (with a transparent setup algorithm) for R1CS over $R_\bullet := R/pR$ with the following efficiency:*

- *round complexity $O(\log^2(M + N))$;*
- *communication complexity $O(\log^2(M + N))$ elements of R/qR;*
- *indexer complexity $O(M + N)$ operations in R/qR;*
- *prover complexity $O(M + N)$ operations in R/qR;*
- *verifier complexity $O(\log^2(M + N))$ operations in R/qR.*

In fact, we construct a preprocessing succinct interactive argument for R1CS based on *leveled bilinear modules*, a new abstraction with multiple instantiations that we deem of independent interest. Theorem 1 follows by instantiating this abstraction using lattices, as we now outline.

An (unleveled) bilinear module [BCS21] consists of modules $M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}$ over a ring R with an R-bilinear map $e \colon M_{\mathrm{L}} \times M_{\mathrm{R}} \to M_{\mathrm{T}}$. Example instantiations include the following.

- **Bilinear groups:** $(R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e) = (\mathbb{F}_p, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_{\mathrm{T}}, e)$, where $|\mathbb{G}_0| = |\mathbb{G}_1| = |\mathbb{G}_{\mathrm{T}}| = p$ and $e \colon \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_{\mathrm{T}}$ is a bilinear (pairing) map.
- **Lattices:** $(R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e) = (R, R, R/q, R/q, e)$, where $R = \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$, q is a large prime, and $e \colon R \times R/q \to R/q$ computes multiplication of ring elements modulo q.

Prior work [BCS21] constructs commitment schemes based on bilinear modules, with messages defined over M_{L}, keys defined over M_{R}, and commitments defined over M_{T}, and gives interactive arguments of knowledge of commitment openings based on the sumcheck protocol. These arguments have linear verification costs in the length of the commitment key, which is the best one can hope for because they are not preprocessing arguments (and so the verifier must receive the long commitment key as input).

In a *leveled* bilinear module, which we introduce, the key space is associated with the message space of another bilinear module.

Definition 2 (informal)**.** *A K-level* **bilinear-module system** *is a collection of K bilinear modules*

$$\{(R, M_{\mathrm{L},i}, M_{\mathrm{R},i}, M_{\mathrm{T},i}, e_i)\}_{i \in [K]}$$

with the same ring R such that $M_{\mathrm{R},i}$ can be "embedded" inside $M_{\mathrm{L},i+1}$ while preserving arithmetic operations (possibly up to some correction factors).

Example instantiations of leveled bilinear modules include the following.

- **Bilinear groups:** $(R, M_{\mathrm{L},i}, M_{\mathrm{R},i}, M_{\mathrm{T},i}, e_i) = (\mathbb{F}_p, \mathbb{G}_{i \bmod 2}, \mathbb{G}_{i+1 \bmod 2}, \mathbb{G}_{\mathrm{T}}, e)$, where where $|\mathbb{G}_0| = |\mathbb{G}_1| = |\mathbb{G}_{\mathrm{T}}| = p$ and $e \colon \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_{\mathrm{T}}$ is a bilinear (pairing) map.
- **Lattices:** $(R, M_{\mathrm{L},i}, M_{\mathrm{R},i}, M_{\mathrm{T},i}, e_i) = (R, R, R/q, R/q, e)$, where $R := \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$ and $e : R \times R/q \to R/q$ computes multiplication of ring elements modulo q. The "embedding map" computes the bit decomposition of ring elements viewed as vectors modulo q: it maps an element of $M_{\mathrm{R},i} := R/q$ viewed as a vector of polynomial coefficients modulo q to $\log q$ elements in $M_{\mathrm{L},i+1} := R$ with coefficients in $\{0, 1\}$.

We use leveled bilinear modules to construct delegation protocols for evaluating polynomials over $M_{\mathrm{L},1}$, which enables achieving succinct verification for commitment openings. In turn, we obtain succinct verification for R1CS from leveled bilinear modules, from which Theorem 1 follows as a special case.

Theorem 2 (informal)**.** *Let $\underline{\mathcal{M}}$ be a leveled bilinear module with $\ell = O(\log(M + N))$ levels, for which the leveled bilinear relation assumption holds. Suppose that $M_{\mathrm{L},1}$ is a ring and I a suitable ideal of $M_{\mathrm{L},1}$. There is a preprocessing interactive argument of knowledge for R1CS over $R_\bullet := M_{\mathrm{L},1}/I \simeq \mathbb{F}^k$ with the following efficiency:*

- *communication complexity $O(\log^2(M + N))$ elements of $M_{\mathrm{T},\ell}$;*
- *round complexity $O(\log^2(M + N))$;*
- *indexer complexity $O(M + N)$ operations in $M_{\mathrm{T},\ell}$ and applications of e_ℓ;*
- *prover complexity $O(M + N)$ operations in $M_{\mathrm{T},\ell}$ and applications of e_ℓ;*
- *verifier complexity $O(\log^2(M + N))$ operations in $M_{\mathrm{T},\ell}$ and applications of e_ℓ.*

The interactive argument in Theorem 2 relies on the *leveled bilinear relation assumption*. This is a falsifiable assumption on leveled bilinear modules implied by the SXDH assumption in the bilinear group instantiation, and by the SIS assumption in the lattice instantiation. For these instantiations, the interactive argument has a transparent (public-coin) setup algorithm.

1.2 Related work

We summarize work on split-and-fold techniques, lattice-based arguments, and Merkle-tree-based arguments.

Split-and-fold techniques over groups. [BCCGP16; BBBPWM18] construct succinct arguments in the discrete logarithm setting, but lack succinct verification. [Lee21] constructs succinct arguments in the bilinear group setting, achieving succinct verification with preprocessing. [BFS20; BHRRS21] construct succinct arguments in the unknown-order group setting, achieving succinct verification without preprocessing (they target uniform computations). Drawing inspiration from [BFS20; BHRRS21] and [Lee21], we achieve succinct verification with preprocessing from an abstract algebraic structure (leveled bilinear modules), which in particular specializes to lattices.

Lattice-based interactive arguments. [BBCPGL18] construct a lattice-based zero-knowledge argument for NP with sublinear (specifically, square-root) communication complexity. [BLNS20] use split-and-fold techniques to construct an interactive argument of knowledge for commitment openings with polylogarithmic communication complexity; subsequently [AL21] reduced the slackness of the openings. [ACK21; BCS21] extend the approach to work for NP statements. [AL21; ACK21] also provide complete security proofs for protocols in [BLNS20], while [BCS21] shows that split-and-fold techniques are related to the sumcheck protocol [LFKN92]. Our starting point is the protocol of [BCS21]: we construct a delegation protocol (itself also related to the sumcheck protocol) for the expensive computation of the verifier in [BCS21]. Finally,

[BS22] uses a more complex recursive approach to achieve logarithmic proof sizes with concrete estimates of communication complexity in the tens of kilobytes for R1CS instances of size 2^{20}. All the aforementioned lattice-based argument systems lack succinct verification.

Many other works aim to provide concretely efficient arguments for NP statements [NS22] and specialized applications including group/ring signatures and proofs of knowledge for lattice-based commitments [ESSLL19; PLS18; BLS19; PLS19; YAZXYW19; ALS20; ENS20; LNS20; LNS21; ESZ22; LNS22].

Lattice-based non-interactive arguments. Several works construct succinct non-interactive arguments (SNARGs) based on non-falsifiable assumptions (believed to be necessary [GW11]) about lattices. [BISW17; BISW18] construct designated-verifier SNARGs by following a paradigm based on linear PCPs [BCIOP13]. These works were subsequently optimized [GMNO18; Nit19; ISW21], and a similar approach was used to obtain public-verifier SNARGs [ACLMT22]. All of these works rely on a private-coin setup algorithm that samples a structured reference string with a trapdoor. This line of work is not directly comparable to our results (we construct interactive arguments from falsifiable assumptions, and moreover the bilinear group and lattice instantiations of our construction have a public-coin setup algorithm).

Merkle-tree-based interactive arguments. A long line of works [BBHR18; BCGGRS19; BCRSVW19; COS20; LSTW21; GLSTW21; BCL22; RR22; XZS22] constructs preprocessing succinct arguments for general NP statements using the Merkle-tree paradigm. These works offer transparent setup and succinct verification with preprocessing. While some of these proof systems offer benefits such as reduced prover complexity in theory [BCL22; RR22] and practice [LSTW21; GLSTW21; XZS22], the communication complexity of these arguments is at present larger than split-and-fold-based proof systems built from classical assumptions (e.g., [BBBPWM18]), which offers communication complexity on the order of a few kilobytes.

2 Techniques

We summarize the main ideas behind our results.

2.1 Our approach

A common approach for constructing succinct arguments is to combine two ingredients: (a) a polynomial interactive oracle proof (PIOP); and (b) a suitable polynomial commitment scheme. PIOPs are information-theoretic proof systems, in which the prover sends polynomials in the form of oracle messages to the verifier, who then performs polynomial evaluation queries to these oracles. The polynomial commitment scheme enables the argument prover to commit to these polynomials and subsequently authenticate answers to queries received from the argument verifier.

The succinct argument that we construct follows this common approach, and our contribution is to achieve a suitable realization of each ingredient. To obtain Theorem 2 it suffices to construct, in the preprocessing model, a PIOP for R1CS with succinct verification (an information-theoretic object) and a polynomial commitment scheme with succinct verification from leveled bilinear modules (a cryptographic object). Below we briefly discuss each ingredient, and we elaborate further on them in later sections; note that, for PIOPs, preprocessing is known as *holography*.

(a) Holographic PIOP for R1CS over product rings. We construct a holographic PIOP for R1CS over product rings $R_\bullet \simeq \mathbb{F}^k$, by extending prior constructions over finite fields $\mathbb{F}$. This is useful because cyclotomic rings commonly employed in lattice cryptography can be expressed as product rings using facts from algebraic number theory. See Section 2.6 for more details.

(b) Polynomial commitment scheme from bilinear modules. Prior constructions of polynomial commitment schemes with succinct verifier based on split-and-fold techniques [BFS20; BHRRS21; Lee21] use delegation protocols and/or preprocessing. We similarly construct a delegation protocol with preprocessing, leveraging an algebraic module-theoretic abstraction called "leveled bilinear modules"; these can be obtained from lattices, for example. Drawing inspiration from [Lee21], this abstraction captures the ability to commit to commitment keys. We explain our construction across several subsections.

- In Section 2.2, we review a polynomial commitment scheme whose proofs of polynomial evaluation, which are based on the sumcheck protocol, have linear-time verification.
- In Section 2.3, we describe a delegation protocol over bilinear groups that reduces verification time to polylogarithmic.
- In Section 2.4, we introduce leveled bilinear modules, and instantiate them using bilinear groups or lattice rings.
- In Section 2.5, we extend the delegation protocol to work over leveled bilinear modules.

Combining. In Section 2.7, we obtain our main result by combining the polynomial commitment scheme with succinct verification and the PIOP over rings.

2.2 Polynomial commitments from sumcheck arguments

Sumcheck arguments [BCS21] are a generalization of the sumcheck protocol and of split-and-fold techniques for proving the correct opening of "sumcheck-friendly" commitments. They are used to construct succinct interactive arguments for NP over an abstract algebraic structure, which can be instantiated with lattices. This gives a succinct interactive argument for NP that exploits the structure of lattice-based commitment schemes.

Sumcheck arguments reduce the task of proving knowledge of a commitment opening to the task of evaluating a polynomial whose coefficients are derived from the commitment key. The verifier has access to the commitment key and can perform this evaluation on its own. The commitment key, however, has linear size, leading to linear verification time.

We now describe how to obtain *polynomial commitment schemes* from sumcheck arguments. We restrict our attention to deterministic commitment schemes (without a hiding property) because these suffice for (non-zero-knowledge) interactive arguments. First, we present the necessary background related to the sumcheck protocol. Then, we focus on sumcheck arguments defined over finite fields $\mathbb{F}$ and discrete logarithm groups $\mathbb{G}$ of prime order. Finally, we discuss sumcheck arguments defined over bilinear modules, an abstract mathematical structure that we will use to express pairing and lattice-based commitments.

Sumcheck protocol. The prover wants to convince the verifier that a given ℓ-variate polynomial P sums to τ over the hypercube $\mathcal{H}^\ell$. While the sumcheck protocol [LFKN92] was introduced for polynomials over fields, it directly extends to work with polynomials over *modules* as we describe below. The following construction is a reduction from the claim $\sum_{\underline{\omega} \in \mathcal{H}^\ell} P(\underline{\omega}) = \tau$ to a claim of the form $P(\underline{r}) = v$.

Protocol 1: sumcheck protocol

The prover P_{SC} and the verifier V_{SC} receive an instance $\mathbb{x}_{\text{SC}} = (R, M, \mathcal{H}, \ell, \tau, \mathcal{C})$, where

- R is a ring,
- M is a module over R,
- $\mathcal{H}$ is a subset of R,
- ℓ is a number of variables,
- $\tau \in M$ is a claimed sum, and
- $\mathcal{C} \subseteq R$ is a sampling set (more about this below).

The prover P_{SC} additionally receives a polynomial $P \in M[X_1, \ldots, X_\ell]$ such that $\sum_{\underline{\omega} \in \mathcal{H}^\ell} P(\underline{\omega}) = \tau$. The protocol has ℓ rounds; in each round the prover sends a univariate polynomial $Q_i(X_i)$ and the verifier responds with a challenge r_i.

1. For $i = 1, \ldots, \ell$:
   (a) P_{SC} sends to V_{SC} the polynomial
   $$Q_i(X_i) := \sum_{\omega_{i+1}, \ldots, \omega_\ell \in \mathcal{H}} P(r_1, \ldots, r_{i-1}, X_i, \omega_{i+1}, \ldots, \omega_\ell) \in M[X_i];$$
   (b) V_{SC} sends to P_{SC} a random challenge $r_i \leftarrow \mathcal{C}$.
2. V_{SC} checks that $\sum_{\omega_1 \in \mathcal{H}} Q_1(\omega_1) = \tau$ and, for $i \in \{2, \ldots, \ell\}$, that $\sum_{\omega_i \in \mathcal{H}} Q_i(\omega_i) = Q_{i-1}(r_{i-1})$.
3. If the checks pass, then V_{SC} sets $v := Q_\ell(r_\ell) \in M$ and outputs the tuple $((r_1, \ldots, r_\ell), v)$.

If $\sum_{\underline{\omega} \in \mathcal{H}^\ell} P(\underline{\omega}) = \tau$, then at the end of Protocol 1, the verifier V_{SC} will always output $((r_1, \ldots, r_\ell), v)$ satisfying $P(r_1, \ldots, r_\ell) = v$. On the other hand, if $\sum_{\underline{\omega} \in \mathcal{H}^\ell} P(\underline{\omega}) \neq \tau$, then for any malicious prover $\tilde{P_{\text{SC}}}$, the verifier's output will only satisfy $P(r_1, \ldots, r_\ell) = v$ with probability at most $\frac{\ell \deg(P)}{|\mathcal{C}|}$. This follows from a strengthening of the analysis of the sumcheck protocol over finite fields, relying on the additional requirement that $\mathcal{C}$ is a "sampling set", which guarantees that non-zero polynomials of a given degree d have at most d roots. The sumcheck protocol over modules is discussed further in [BCS21].

Polynomial commitment scheme. A polynomial commitment scheme enables a prover to commit to a polynomial and later prove that a claimed polynomial evaluation at a given point is correct. For concreteness, we consider multilinear polynomials whose coefficients are defined by a vector of elements as follows.

Definition 2.1. *We index the entries of a vector $\underline{v}$ of length $n = 2^\ell$ via binary strings $(i_1, \dots, i_\ell) \in \{0,1\}^\ell$, and define the corresponding multilinear polynomial*

$$p_{\underline{v}}(X_1, \dots, X_\ell) := \sum_{i_1,\dots,i_\ell \in \{0,1\}} X_1^{i_1} \cdots X_\ell^{i_\ell} \cdot v_{i_1,\dots,i_\ell} .$$

We describe a polynomial commitment scheme based on Pedersen commitments for committing to the polynomial $p_{\underline{\mathsf{m}}}(X_1, \dots, X_{\log n})$, where $\underline{\mathsf{m}} \in \mathbb{F}^n$ and $\mathbb{F}$ is a finite field of prime order p. The commitment is an element of a group $\mathbb{G}$ of order p. In the proof of polynomial evaluation, the prover wishes to convince the verifier of the following $\mathcal{NP}$ statement:

Task 1. *Given a commitment $\mathsf{C} \in \mathbb{G}$, a commitment key $\underline{\mathsf{G}} \in \mathbb{G}^n$, an evaluation point $\underline{z} \in \mathbb{F}^{\log n}$, and a claimed evaluation $u \in \mathbb{F}$, prove knowledge of the polynomial $p_{\underline{\mathsf{m}}}$ (i.e., of the coefficients $\underline{\mathsf{m}} \in \mathbb{F}^n$) such that $p_{\underline{\mathsf{m}}}(\underline{z}) = u$ and $\mathsf{C} = \langle \underline{\mathsf{m}}, \underline{\mathsf{G}} \rangle$.*

Using Definition 2.1 we define the polynomial $p_{\underline{\mathsf{G}}}(X_1, \dots, X_{\log n})$. Here, $p_{\underline{\mathsf{G}}}(\underline{X})$ defines a polynomial function $p_{\underline{\mathsf{G}}} \colon \mathbb{F}^{\log n} \to \mathbb{G}$ over $\mathbb{G}$, where addition corresponds to the group operation and multiplication with an element in $\mathbb{F}$ corresponds to scalar multiplication with the same element. Observe that $\sum_{\underline{\omega} \in \{-1,1\}^\ell} p_{\underline{\mathsf{m}}}(\underline{\omega}) p_{\underline{\mathsf{G}}}(\underline{\omega}) = 2^\ell \cdot \mathsf{C}$.

Protocol 2 is a succinct interactive argument for Task 1 based on a sumcheck argument. The only non-succinct verifier operation is colored blue.

Protocol 2: sumcheck argument for polynomial evaluation

For $n = 2^\ell$, the prover and verifier receive as input a commitment key $\underline{\mathsf{G}} \in \mathbb{G}^n$, a commitment $\mathsf{C} \in \mathbb{G}$, an evaluation point $\underline{z} := (z_1, z_2, \dots, z_\ell) \in \mathbb{F}^\ell$, and a claimed evaluation $u \in \mathbb{F}$. The prover also receives as input an opening $\underline{\mathsf{m}} \in \mathbb{F}^n$ such that $\mathsf{C} = \langle \underline{\mathsf{m}}, \underline{\mathsf{G}} \rangle$.

The prover and verifier engage in a sumcheck protocol for the claim

$$\sum_{\underline{\omega} \in \{-1,1\}^\ell} P'(\underline{\omega}) = 2^\ell \cdot (\mathsf{C}, u) \ ,$$

where $P'(\underline{X}) := (p_{\underline{\mathsf{m}}}(\underline{X}) \cdot p_{\underline{\mathsf{G}}}(\underline{X}), p_{\underline{\mathsf{m}}}(\underline{X}) \cdot p_{\underline{\tilde{z}}}(\underline{X}))$ and $\underline{\tilde{z}} := \bigotimes_{i=1}^{\ell} (1, z_i) = (1, z_1, z_2, z_1 z_2, \dots, z_1 \cdots z_\ell)$. As defined in Protocol 1, the sumcheck protocol uses the instance

$$\mathbb{x}_{\text{SC}} := (R = \mathbb{F},\ M = \mathbb{G} \times \mathbb{F},\ \mathcal{H} = \{-1,1\},\ \ell = \log n,\ \tau = 2^\ell \cdot (\mathsf{C}, u),\ \mathcal{C} = \mathbb{F}) \ ,$$

and the prover additionally knows the polynomial $P'(\underline{X}) \in (\mathbb{G} \times \mathbb{F})[\underline{X}]$.

After the end of the sumcheck protocol, if the verifier's checks pass, the prover learns the randomness $\underline{r} \in \mathbb{F}^\ell$ used in the protocol, and the verifier learns $(\underline{r}, v) \in \mathbb{F}^\ell \times \mathbb{F}$. Then, the prover computes and sends $w := p_{\underline{\mathsf{m}}}(\underline{r}) \in \mathbb{F}$; the verifier computes $p_{\underline{\mathsf{G}}}(\underline{r}) \in \mathbb{G}$ and $p_{\underline{\tilde{z}}}(\underline{r}) \in \mathbb{F}$ and checks that $(w \cdot p_{\underline{\mathsf{G}}}(\underline{r}), w \cdot p_{\underline{\tilde{z}}}(\underline{r})) = v$.

The task to delegate. The only expensive operation that the verifier has to compute is the final multilinear polynomial evaluation $p_{\underline{\mathsf{G}}}(\underline{r})$; because $\underline{\tilde{z}} := \bigotimes_{i=1}^{\ell}(1, z_i)$, it holds that $p_{\underline{\tilde{z}}}(\underline{r}) = \prod_{i=1}^{\ell}(1 + r_i z_i)$ which can be evaluated in $O(\ell) = O(\log n)$ operations. Our goal is to reduce the verifier complexity by delegating the polynomial evaluation $p_{\underline{\mathsf{G}}}(\underline{r})$ to the prover. This means that the prover sends $\mathsf{V} \in \mathbb{G}$ and has to prove the following $\mathcal{P}$ statement to the verifier.

Task 2. *Given a commitment key* $\underline{\mathsf{G}} \in \mathbb{G}^n$*, an evaluation point* $\underline{r} \in \mathbb{F}^{\log n}$*, and a claimed evaluation* $\mathsf{V} \in \mathbb{G}$*, prove that* $p_{\underline{\mathsf{G}}}(\underline{r}) = \mathsf{V}$.

It is not known how to delegate this task over finite fields $\mathbb{F}$ and discrete logarithm groups $\mathbb{G}$ of prime order. However, we will show a delegation protocol for bilinear groups and lattices. First, we define bilinear modules, an algebraic abstraction that allows us to instantiate Protocol 2 in these settings.

Generalization to bilinear modules. We need the commitment scheme and sumcheck argument above to work over more general algebraic structures, specifically over bilinear modules. A bilinear module $\mathsf{BM} = (R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e)$ consists of a ring R, three R-modules $M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}$, and an R-bilinear map $e\colon M_{\mathrm{L}} \times M_{\mathrm{R}} \to M_{\mathrm{T}}$.

In a *generalized Pedersen commitment* over a bilinear module BM, the commitment key is a random vector $\underline{\mathsf{G}} \in M_{\mathrm{R}}^n$ and the commitment to the message $\underline{\mathsf{m}} \in M_{\mathrm{L}}^n$ is $\mathsf{C} := \langle \underline{\mathsf{m}}, \underline{\mathsf{G}} \rangle := \sum_{i=1}^{n} e(\mathsf{m}_i, \mathsf{G}_i) \in M_{\mathrm{T}}$. The commitment scheme is binding for messages of *bounded norm* if given a random vector $\underline{\mathsf{G}} \in M_{\mathrm{R}}^n$, it is hard to find $\underline{\mathsf{m}} \in M_{\mathrm{L}}^n$ with $\underline{\mathsf{m}} \neq 0$ and $\|\underline{\mathsf{m}}\| \leq B_{\mathsf{c}}$ such that $\langle \underline{\mathsf{m}}, \underline{\mathsf{G}} \rangle = 0$. We call this assumption *bilinear relation assumption*.

The generalized Protocol 2 works exactly as before, except for a new check on the norm of w to guarantee that the commitment opening is binding.

In the case of discrete logarithm groups, which is used in Protocol 2, we have $(R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e) := (\mathbb{F}, \mathbb{F}, \mathbb{G}, \mathbb{G}, e)$, using group exponentiation for e. Other instantiations of bilinear modules include bilinear groups and ideal lattices. In the bilinear group setting, $(R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e) := (\mathbb{F}, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_{\mathrm{T}}, e)$ using the bilinear (pairing) operation for e. In the lattice setting, $(R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e) := (R, R, R/qR, R/qR, \times)$, where $R := \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$, Φ_d is the d-th cyclotomic polynomial and $\times$ is polynomial multiplication modulo q. The bilinear relation assumption for the three instantiations corresponds to discrete logarithm, double pairing, and SIS assumptions respectively. In the discrete logarithm and the bilinear group setting, the underlying norm is such that all non-zero elements have norm 1, whereas in the ideal lattice setting we consider the ℓ_∞-norm.

2.3 Warmup: delegation over bilinear groups

Consider the setting of bilinear groups: there are three groups $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_{\mathrm{T}}$ of prime size p and a bilinear map $e\colon \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_{\mathrm{T}}$. When the polynomial commitment scheme and sumcheck argument from Section 2.2 are realized over this instantiation of bilinear modules, Task 2 becomes the following.

Task 3. *Given a commitment key* $\underline{\mathsf{G}} \in \mathbb{G}_1^n$*, an evaluation point* $\underline{r} \in \mathbb{F}^{\log n}$*, and a claimed evaluation* $\mathsf{V} \in \mathbb{G}_1$*, prove that* $p_{\underline{\mathsf{G}}}(\underline{r}) = \mathsf{V}$.

We describe an interactive proof with succinct verification for this task that is based on techniques from [Lee21] (and variants [Tha22]). Below we review the main ideas behind these techniques, and then discuss the challenges that arise in extending them to work for more general algebraic structures.

Review: delegation ideas from [Lee21]. Consider an additional polynomial commitment scheme whose message space is $\mathbb{G}_1^n$ and whose key space is $\mathbb{G}_0^n$:

- a commitment key is a random $\underline{\mathsf{H}} \in \mathbb{G}_0^n$;
- a message is $\underline{\mathsf{G}} \in \mathbb{G}_1^n$ (which can be the commitment key from Task 3);
- $\mathsf{C}' := \langle \underline{\mathsf{H}}, \underline{\mathsf{G}} \rangle = \sum_{i=1}^n e(\mathsf{H}_i, \mathsf{G}_i)$ is a commitment to $\underline{\mathsf{G}}$ using key $\underline{\mathsf{H}}$.

Since $\underline{\mathsf{G}}$ and $\underline{\mathsf{H}}$ are sampled during the setup phase, C' can be computed during a preprocessing phase. Then, Task 3 can be replaced by the following task.

Task 4. *Given a commitment $\mathsf{C}' = \langle \underline{\mathsf{H}}, \underline{\mathsf{G}} \rangle \in \mathbb{G}_\mathrm{T}$ computed in a preprocessing phase by the (honest) indexer, an evaluation point $\underline{r} \in \mathbb{F}^{\log n}$, and a claimed evaluation $\mathsf{V} \in \mathbb{G}_1$, prove that $p_{\underline{\mathsf{G}}}(\underline{r}) = \mathsf{V}$.*

This opens up the possibility of succinct verification because the verifier receives as input $\mathsf{C}' \in \mathbb{G}_\mathrm{T}$ rather than $\underline{\mathsf{G}} \in \mathbb{G}_1^n$. In fact, Task 4 is similar to the original task (Task 1) defined in the setting of bilinear groups. A difference is that in Task 1 the verifier is also given the commitment key. However, to achieve succinct verification the verifier here cannot receive $\underline{\mathsf{H}} \in \mathbb{G}_0^n$ as input.

Reducing the key size. With further ideas from [Lee21], one can reduce to a *smaller* commitment key over $\mathbb{G}_0^{n/2}$, and then apply the same technique with the roles of $\mathbb{G}_0$ and $\mathbb{G}_1$ reversed. One can repeat this until the verifier need only perform a computation on a constant-size commitment key.

Instead of committing to $\underline{\mathsf{G}}$ using a commitment key of length n, split $\underline{\mathsf{G}}$ into two halves: $\underline{\mathsf{G}} := (\underline{\mathsf{G}}[L], \underline{\mathsf{G}}[R]) \in \mathbb{G}_1^{n/2} \times \mathbb{G}_1^{n/2}$. During the preprocessing phase, the indexer computes the commitments $\mathsf{C}_L := \langle \underline{\mathsf{H}}, \underline{\mathsf{G}}[L] \rangle \in \mathbb{G}_\mathrm{T}$ and $\mathsf{C}_R := \langle \underline{\mathsf{H}}, \underline{\mathsf{G}}[R] \rangle \in \mathbb{G}_\mathrm{T}$ using the commitment key $\underline{\mathsf{H}} \in \mathbb{G}_0^{n/2}$.

Instead of C', which is a commitment to $\underline{\mathsf{G}}$, the verifier now has C_L and C_R, so we can no longer apply the sumcheck argument for polynomial evaluation (Protocol 2) to Task 4. To remedy this, we use the fact that the verifier can compute a commitment to any linear combination of $\underline{\mathsf{G}}[L]$ and $\underline{\mathsf{G}}[R]$. Then, it suffices to find a linear combination $\underline{\mathsf{G}}' \in \mathbb{G}_1^{n/2}$ and an evaluation point $\underline{r}' \in \mathbb{F}^{\log n - 1}$ such that $p_{\underline{\mathsf{G}}}(\underline{r}) = p_{\underline{\mathsf{G}}'}(\underline{r}')$.

From Definition 2.1, $p_{\underline{\mathsf{G}}}(\underline{X}) := \sum_{i_1,\dots,i_{\log n} \in \{0,1\}} X_1^{i_1} \cdots X_\ell^{i_{\log n}} \cdot \mathsf{G}_{i_1,\dots,i_{\log n}}$ where $\underline{\mathsf{G}} := (\mathsf{G}_1, \dots, \mathsf{G}_n)$. Hence, $p_{\underline{\mathsf{G}}}(\underline{X}) = p_{\underline{\mathsf{G}}[L] + X_1 \underline{\mathsf{G}}[R]}(X_2, \dots, X_{\log n})$ and Task 4 reduces to the following task.

Task 5. *Given a commitment $\mathsf{C}' := \mathsf{C}_L + r_1 \mathsf{C}_R$, where $\mathsf{C}_L := \langle \underline{\mathsf{H}}, \underline{\mathsf{G}}[L] \rangle \in \mathbb{G}_\mathrm{T}$ and $\mathsf{C}_R = \langle \underline{\mathsf{H}}, \underline{\mathsf{G}}[R] \rangle \in \mathbb{G}_\mathrm{T}$ are computed in a preprocessing phase, an evaluation point $\underline{r}' \in \mathbb{F}^{\log n - 1}$, and a claimed evaluation $\mathsf{V} \in \mathbb{G}_1$, prove that $p_{\underline{\mathsf{G}}'}(\underline{r}') = \mathsf{V}$, where $\underline{\mathsf{G}}' := \underline{\mathsf{G}}[L] + r_1 \underline{\mathsf{G}}[R] \in \mathbb{G}_1^{n/2}$.*

Challenge: what happens over bilinear modules? The ideas described above work over bilinear groups due to two fortunate coincidences.

- There are two bilinear modules $(\mathbb{F}, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_\mathrm{T}, e)$ and $(\mathbb{F}, \mathbb{G}_1, \mathbb{G}_0, \mathbb{G}_\mathrm{T}, e)$ that lead to two commitment schemes *with opposite message space and key space*.
- The output claim produced by a sumcheck argument over the first bilinear module is a claim that can be proved using a sumcheck argument over the second bilinear module, and vice versa.

Unfortunately, the situation with general bilinear modules is not so straightforward. Even if the first property is satisfied (namely, both $\mathsf{BM}_1 = (R, M_\mathrm{L}, M_\mathrm{R}, M_\mathrm{T}, e)$ and $\mathsf{BM}_2 = (R, M_\mathrm{R}, M_\mathrm{L}, M_\mathrm{T}, e)$ are bilinear modules), the second property is not. Since $\underline{\mathsf{G}} \in M_\mathrm{R}^n$ is *random* (so to act as a commitment key over BM_1), $\underline{\mathsf{G}}$ may not have *bounded norm*. The norm bound is required in order to make a binding commitment to $\underline{\mathsf{G}}$, when it acts as a message for BM_2! *This precludes using the same repeated reduction idea over* BM_1 *and* BM_2.

2.4 Leveled bilinear modules

In order to build a delegation protocol for general bilinear modules and prove Theorem 2 (and thus Theorem 1), we want the ability to commit to commitment keys from successive reductions using new bilinear modules.

To this end, we consider *multiple levels* of compatible bilinear modules, capable of mapping statements about commitment keys for "lower-level" commitment schemes to statements about messages in "higher-level" commitment schemes. We formalize this new abstract algebraic structure and call it a *leveled bilinear module system*. We also give post-quantum instantiations based on ideal lattices.

Defining leveled bilinear modules. A K-level bilinear module system is a list of K bilinear module systems over the *same* ring R, each satisfying the bilinear relation assumption:

$$\{\mathsf{BM}_i\}_{i\in[K]} = \{(R, M_{\mathrm{L},i}, M_{\mathrm{R},i}, M_{\mathrm{T},i}, e_i)\}_{i\in[K]} \enspace .$$

Further, to allow commitments to Pedersen commitment keys, successive levels are connected by two maps:

- an *upward map* $\mathsf{up}_i \colon M_{\mathrm{R},i} \to M_{\mathrm{L},i+1}^{\delta_{i+1}}$ that lifts keys at level i to δ_{i+1} small-norm messages at level $i+1$; and
- a *downward map* $\mathsf{dn}_i \colon M_{\mathrm{L},i+1}^{\delta_{i+1}} \to M_{\mathrm{R},i}$ that projects messages at level $i+1$ to keys at level i.

The two maps up_i and dn_i cancel each other out: $\mathsf{dn}_i \circ \mathsf{up}_i$ is the identity map on $M_{\mathrm{R},i}$. Messages produced by up_i are within the binding space of the commitment scheme at level $i+1$. For each level $i \in [K-1]$, the upward map up_i (and hence also dn_i) must satisfy some homomorphic properties:

- for every $m_1, m_2 \in M_{\mathrm{R},i}$, $\mathsf{up}_i(m_1 + m_2) = \mathsf{up}_i(m_1) + \mathsf{up}_i(m_2) \bmod \ker \mathsf{dn}_i$;
- for every $r \in R$ and $m \in M_{\mathrm{R},i}$, $\mathsf{up}_i(r \cdot m) = r \cdot \mathsf{up}_i(m) \bmod \ker \mathsf{dn}_i$.

In fact, these conditions imply that $M_{\mathrm{R},i}$ and $M_{\mathrm{L},i+1}^{\delta_{i+1}} / \ker \mathsf{dn}_i$ are isomorphic as R-modules via up_i and dn_i. Note that if "$\bmod \ker \mathsf{dn}_i$" was removed from the two conditions above, then $M_{\mathrm{R},i}$ and $M_{\mathrm{L},i+1}^{\delta_{i+1}}$ would be isomorphic as R-modules. This would be too rigid for lattice instantiations, in which for every $i \in [K-1]$ the upward map up_i takes statements about commitment keys modulo a prime q to multiple statements about integers of bounded norm, which can be messages for higher-level commitment schemes. Also, equations modulo q may not hold exactly over the integers, and working $\bmod \ker \mathsf{dn}_i$ allows for correction factors.

Using up_i, claims about polynomial evaluations over commitment key elements can be lifted from $M_{\mathrm{R},i}$ to $M_{\mathrm{L},i+1}$ to act as inputs for proof systems over BM_{i+1}. Conversely, using dn_i, statements proved about lifted polynomial evaluations reduce to similar statements about polynomial evaluations over the commitment keys. Leveled bilinear module systems neatly encapsulate the algebraic requirements for interactive arguments like [Lee21], and facilitate extending those ideas to other cryptographic settings.

Instantiations. We describe three instantiations of leveled bilinear-module systems.

- A "2-cycle" based on bilinear groups. Given a bilinear group $(\mathbb{F}, \mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_\mathrm{T}, e)$, we set $M_{\mathrm{L},i} := \mathbb{G}_{i \bmod 2}$, $M_{\mathrm{R},i} := \mathbb{G}_{i+1 \bmod 2}$, $M_{\mathrm{T},i} := \mathbb{G}_\mathrm{T}$, $\delta_i = 1$, and $e_i := e$. Hence $M_{\mathrm{R},i}$ and $M_{\mathrm{L},i+1}$ are equal. For each level $i \in [K-1]$, the upward map $\mathsf{up}_i \colon \mathbb{G}_{i \bmod 2} \to \mathbb{G}_{i+1 \bmod 2}$ and downward map $\mathsf{dn}_i \colon \mathbb{G}_{i+1 \bmod 2} \to \mathbb{G}_{i \bmod 2}$ are the identity map. At each level, the bilinear relation assumption is implied by the SXDH assumption. This instantiation works for any number of levels.

- A first instantiation based on ideal lattices. Let d be a prime power, $\Phi_d(X)$ the d-th cyclotomic polynomial, $R = \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$ the corresponding cyclotomic ring, and $q_1, \dots, q_K \in \mathbb{N}$. Let $M_{\mathrm{L},i} := R$, $M_{\mathrm{R},i} := R/q_iR$, $M_{\mathrm{T},i} := R/q_iR$, and e_i be the multiplication of ring elements modulo q_i.

  We "lift" an element m of $M_{\mathrm{R},i} = R/q_iR$ to an element of $M_{\mathrm{L},i+1} = \mathbb{Z}[X]/\langle X^d + 1 \rangle$ with norm at most q_i by viewing it as a polynomial over the integers rather than modulo q_i. For each level $i \in [K-1]$, the upward map $\mathsf{up}_i \colon R/qR \to R$ lifts polynomials modulo q to integer polynomials, and the downward map

$\mathsf{dn}_i \colon R \to R/q_iR$ performs the reverse operation, i.e., reduction modulo q_i. At each level, the bilinear relation assumption follows from the ring SIS assumption modulo q_i.

Unfortunately, this first instantiation is somewhat inefficient, and insecure when K is super-constant. This is because in order for the ring SIS assumption modulo q_i to be hard with respect to messages of norm up to q_{i-1}, we require $q_i \gg q_{i-1}$, so that $q_K \gg \cdots \gg q_1$. Moreover, based on the parameters required by the proof system that we use, the gap between each modulus can force q_K to be exponentially large when $K = \omega(1)$, which poses problems for the hardness of ring SIS.

This motives the following improved instantiation.

- A "1-cycle" based on ideal lattices. Let d be a prime power, $\Phi_d(X)$ the d-th cyclotomic polynomial, $R = \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$ the corresponding cyclotomic ring, and $q \in \mathbb{N}$. Let $M_{\mathrm{L},i} := R$, $M_{\mathrm{R},i} := R/qR$, $M_{\mathrm{T},i} := R/qR$, and e_i be the multiplication of ring elements modulo q.

  An element in R can be viewed as a polynomial with d coefficients. We "lift" an element m of $M_{\mathrm{R},i} = R/qR$ to $\log q$ elements of $M_{\mathrm{L},i+1} = \mathbb{Z}[X]/\langle X^d + 1 \rangle$ with norm at most 1 by computing the bit decomposition of the coefficients of m. For each level $i \in [K-1]$, the upward map $\mathsf{up}_i \colon R/qR \to R$ lifts polynomials modulo q to integer polynomials using bit decomposition, and the downward map $\mathsf{dn}_i \colon R \to R/qR$ performs the reverse operation, i.e., bit composition modulo q. At each level, the bilinear relation assumption follows from the ring SIS assumption modulo q. This instantiation works for any number of levels.

2.4.1 Comparison with prior algebraic structures

Tiered commitment schemes. Some prior works also use leveled algebraic structures to construct argument systems. [Gro11] constructs two-tiered commitment schemes, in which commitments in $\mathbb{G}_0$ (to messages in $\mathbb{F}$) are themselves treated as messages and used to produce "commitments to commitments" in $\mathbb{G}_{\mathrm{T}}$. [BLNS20] uses a lattice construction to "commit to commitments" over multiple levels. In contrast to our work, the focus in these works is committing to commitments, which would lead to an abstraction that is different from ours ($M_{\mathrm{T},i}$, rather than $M_{\mathrm{R},i}$, is identified with $M_{\mathrm{L},i+1}$).[1]

Graded encodings (a.k.a. multilinear maps). Leveled modules may be reminiscent of graded encoding schemes, in which elements of groups can be multiplied together up to a certain number of multiplications. We explain the main differences between graded encoding schemes and leveled bilinear-module systems.

Graded encodings of different levels usually consist of elements of the same ring, with homomorphic properties when combining encodings at different levels. By contrast, leveled bilinear modules feature different modules at each level, and the embedding maps between levels do not fully preserve homomorphism. This means that only objects at the same level can be multiplied together, and since homomorphism is limited, leveled bilinear modules cannot be used to construct a multilinear map.

Constructions of graded encoding schemes typically rely on lattice assumptions [GGH13; LSS14; GGH15] or integer assumptions (e.g., the approximate GCD problem) [CLT13; CLT15; MZ18] that have been subject to many attacks [Cor+15; CFLMR16; CLLT16; CLLT17]. By contrast, we give comparatively simple instantiations of leveled bilinear modules based on bilinear groups and ideal lattices, providing the relevant security properties under standard cryptographic assumptions (SXDH and SIS respectively).

[1]Of course, in our lattice instantiation, $M_{\mathrm{R},i}$ and $M_{\mathrm{T},i}$ happen to be the same.

2.5 Delegation over leveled bilinear-module systems

The polynomial commitment scheme and the sumcheck argument from Section 2.2 can be defined over a bilinear module, and in particular over the first level of a leveled bilinear-module system. In this case, the prover's goal is to convince the verifier of the following $\mathcal{NP}$ statement.

Task 6. *Given a commitment* $\mathsf{C} \in M_{\mathrm{T},1}$, *a commitment key* $\underline{\mathsf{G}} \in M_{\mathrm{R},1}^n$, *an evaluation point* $\underline{z} \in R^{\log n}$, *and a claimed evaluation* $u \in M_{\mathrm{L},1}$, *prove knowledge of* $\underline{\mathsf{m}} \in M_{\mathrm{L},1}^n$ *such that* $p_{\underline{\mathsf{m}}}(\underline{z}) = u$ *and* $\mathsf{C} = \langle \underline{\mathsf{m}}, \underline{\mathsf{G}} \rangle$.

The succinct interactive protocol for the above task is a generalization of Protocol 2 over bilinear modules. Even though for certain settings (e.g., lattices) norm manipulations and selecting appropriate challenge spaces $\mathcal{C} \subseteq R$ are important, for simplicity in this overview we ignore these issues.

Protocol 3: sumcheck argument for polynomial evaluation over $\underline{\mathcal{M}}$

For $n = 2^\ell$, the prover and verifier receive as input a commitment key $\underline{\mathsf{G}} \in M_{\mathrm{R},1}^n$, a commitment $\mathsf{C} \in M_{\mathrm{T},1}$, an evaluation point $\underline{z} := (z_1, z_2, \ldots, z_\ell) \in R^\ell$, and a claimed evaluation $u \in M_{\mathrm{L},1}$. The prover also receives as input an opening $\underline{\mathsf{m}} \in M_{\mathrm{L},1}^n$ such that $\mathsf{C} = \langle \underline{\mathsf{m}}, \underline{\mathsf{G}} \rangle$.

The prover and verifier engage in a sumcheck protocol for the claim

$$\sum_{\underline{\omega} \in \{-1,1\}^\ell} P'(\underline{\omega}) = 2^\ell \cdot (\mathsf{C}, u),$$

where $P'(\underline{X}) := (p_{\underline{\mathsf{m}}}(\underline{X}) \cdot p_{\underline{\mathsf{G}}}(\underline{X}), p_{\underline{\mathsf{m}}}(\underline{X}) \cdot p_{\underline{\tilde{z}}}(\underline{X}))$ and $\underline{\tilde{z}} := \bigotimes_{i=1}^{\ell} (1, z_i) = (1, z_1, z_2, z_1 z_2, \ldots, z_1 \cdots z_\ell)$. As defined in Protocol 1, the sumcheck protocol uses the instance

$$\mathbb{x}_{\mathrm{SC}} := (R,\, M = M_{\mathrm{T},1} \times M_{\mathrm{L},1},\, \mathcal{H} = \{-1, 1\},\, \ell = \log n,\, \tau = 2^\ell \cdot (\mathsf{C}, u),\, \mathcal{C} \subseteq R) \ ,$$

and the prover additionally knows the polynomial $P'(\underline{X}) \in (M_{\mathrm{T},1} \times M_{\mathrm{L},1})[\underline{X}]$.

After the end of the sumcheck protocol, if the verifier's checks pass, the prover learns the randomness $\underline{r} \in \mathcal{C}^\ell$ used in the protocol, and the verifier learns $(\underline{r}, v) \in \mathcal{C}^\ell \times (M_{\mathrm{T},1} \times M_{\mathrm{L},1})$. Then, the prover computes and sends $w := p_{\underline{\mathsf{m}}}(\underline{r}) \in M_{\mathrm{L},1}$; the verifier computes $p_{\underline{\mathsf{G}}}(\underline{r}) \in M_{\mathrm{R},1}$ and $p_{\underline{\tilde{z}}}(\underline{r}) \in R$ and checks that $(w \cdot p_{\underline{\mathsf{G}}}(\underline{r}), w \cdot p_{\underline{\tilde{z}}}(\underline{r})) = v$.

Delegation using the leveled bilinear-module system. The above protocol reduces proving that $p_{\underline{\mathsf{m}}}(\underline{z}) = u \in M_{\mathrm{L},1}$ to checking the polynomial evaluation $p_{\underline{\mathsf{G}}}(\underline{r}) = \mathsf{V} \in M_{\mathrm{R},1}$. Using the maps of the leveled bilinear-module system, we compute $\mathsf{up}_1(\underline{\mathsf{G}}) \in (M_{\mathrm{L},2}^{\delta_2})^n$, where up_1 is applied to each coordinate of $\underline{\mathsf{G}}$, and $\mathsf{V}' \equiv \mathsf{up}_1(\mathsf{V}) \bmod \ker(\mathsf{dn}_1) \in M_{\mathrm{L},2}^{\delta_2}$. Then, we transform the evaluation $p_{\underline{\mathsf{G}}}(\underline{r}) = \mathsf{V} \in M_{\mathrm{R},1}$ to δ_2 evaluations over $M_{\mathrm{L},2}$:

$$p_{\mathsf{up}_1(\underline{\mathsf{G}})}(\underline{r}) = \mathsf{V}' \ .$$

The function up_1 maps an element in $M_{\mathrm{R},1}$ to multiple elements in $M_{\mathrm{L},2}$. We reduce to a single element of $M_{\mathrm{L},2}$ by computing a random linear combination using challenges sent by the verifier. For the rest of this section, we ignore this issue and focus on the case where up_i maps an element of an $M_{\mathrm{R},i}$ to a *single* element of $M_{\mathrm{L},i+1}$ (i.e., $\delta_{i+1} = 1$).

We can apply the key reduction idea presented in Section 2.3 to reduce to a statement of smaller size. During the preprocessing phase, the indexer computes the commitments $\mathsf{C}_L = \langle \mathsf{up}_1(\underline{\mathsf{G}}[L]), \underline{\mathsf{H}} \rangle \in M_{\mathrm{T},2}$ and $\mathsf{C}_R = \langle \mathsf{up}_1(\underline{\mathsf{G}}[R]), \underline{\mathsf{H}} \rangle \in M_{\mathrm{T},2}$, where $\underline{\mathsf{G}} := (\underline{\mathsf{G}}[L], \underline{\mathsf{G}}[R]) \in M_{\mathrm{R},1}^{n/2} \times M_{\mathrm{R},1}^{n/2}$. Task 6 reduces to the following.

Task 7. *Given a commitment $\mathsf{C}' := \mathsf{C}_L + r_1\mathsf{C}_R$, where $\mathsf{C}_L := \langle \mathsf{up}_1(\underline{\mathsf{G}}[L]), \underline{\mathsf{H}}\rangle \in M_{\mathrm{T},2}$ and $\mathsf{C}_R = \langle \mathsf{up}_1(\underline{\mathsf{G}}[R]), \underline{\mathsf{H}}\rangle \in M_{\mathrm{T},2}$ are computed in a preprocessing phase, an evaluation point $\underline{r}' \in R^{\log n-1}$, and a claimed evaluation $\mathsf{V}' \in M_{\mathrm{L},2}$, prove that $p_{\underline{\mathsf{G}}'}(\underline{r}') = \mathsf{V}'$, where $\underline{\mathsf{G}}' := \mathsf{up}_1(\underline{\mathsf{G}}[L]) + r_1 \cdot \mathsf{up}_1(\underline{\mathsf{G}}[R]) \in M_{\mathrm{L},2}^{n/2}$.*

Final protocol: delegation of polynomial evaluations with succinct verifier. Below we sketch the final protocol. There are $\ell := \log n$ iterations of Protocol 3. In the i-th iteration the instance has size $n/2^i$ and is defined over the i-th level of the leveled bilinear module. After ℓ iterations of Protocol 3, the verifier checks the evaluation of a constant polynomial, which can be done without help from the prover.

Protocol 4: delegation of polynomial evaluations over $\underline{\mathcal{M}}$

Setup. Given an upper bound n on the size of $\underline{\mathsf{m}}$ (the number of polynomial coefficients), the setup algorithm samples a leveled bilinear-module system with $\log n$ levels and commitment keys $\underline{\mathsf{G}}_i \in M_{\mathrm{R},i}^{n/2^{i-1}}$ for $i \in \{1, \ldots, \log n + 1\}$.

Indexer. In a preprocessing phase (i.e., before receiving $\underline{\mathsf{m}}$), the indexer computes

$$\mathsf{C}_{L,i} := \langle \mathsf{up}_i(\underline{\mathsf{G}}_i[L]), \underline{\mathsf{G}}_{i+1}\rangle \in M_{\mathrm{T},i+1} \ \text{, and} \mathsf{C}_{R,i} = \langle \mathsf{up}_i(\underline{\mathsf{G}}_i[R]), \underline{\mathsf{G}}_{i+1}\rangle \in M_{\mathrm{T},i+1}$$

for $i \in \{1, \ldots, \log n\}$. Finally, the indexer sets outputs the proving key $\mathsf{ipk} := (\underline{\mathsf{G}}_i)_{i=1}^{\log n+1}$ and verification key $\mathsf{ivk} := ((\mathsf{C}_{L,i}, \mathsf{C}_{R,i})_{i=1}^{\log n}, \underline{\mathsf{G}}_{\log n})$.

Interactive phase. For $n = 2^\ell$, the prover and verifier receive as input a commitment $\mathsf{C} \in M_{\mathrm{T},1}$, an evaluation point $\underline{z} := (z_1, z_2, \ldots, z_\ell) \in R^\ell$, and a claimed evaluation $u \in M_{\mathrm{L},1}$. The prover also receives as input the proving key ipk and an opening $\underline{\mathsf{m}} \in M_{\mathrm{L},1}^n$ such that $\mathsf{C} = \langle \underline{\mathsf{m}}, \underline{\mathsf{G}}\rangle$. The verifier also receives as input the verification key ivk.

The prover and verifier engage in $\log n$ iterations of Protocol 3. The first iteration reduces the claim $p_{\underline{\mathsf{m}}}(\underline{z}) = u$ to proving that $p_{\underline{\mathsf{G}}_1}(\underline{r}_1) = \mathsf{V}_1$, which can be reduced to the claim $p_{\underline{\mathsf{G}}'_1}(\underline{r}'_1) = \mathsf{V}'_1 \in M_{\mathrm{L},2}$ as in Task 7. Similarly, the i-th iteration reduces the claim $p_{\underline{\mathsf{G}}'_{i-1}}(\underline{r}'_{i-1}) = \mathsf{V}'_{i-1} \in M_{\mathrm{L},i}$ to proving that $p_{\underline{\mathsf{G}}'_i}(\underline{r}'_i) = \mathsf{V}'_i \in M_{\mathrm{L},i+1}$. Finally, the last claim is $p_{\underline{\mathsf{G}}_{\log n}}(\underline{r}_{\log n}) = \mathsf{V}_{\log n}$, which the verifier can check directly using the key $\underline{\mathsf{G}}_{\log n}$.

The indexer performs $O(n)$ operations. Subsequently, the prover and verifier interact over $O(\log^2 n)$ rounds. The communication complexity is $O(\log^2 n)$ elements of the ring and modules of the leveled bilinear-module system: each iteration of the $O(\log n)$ iterations of Protocol 3 has communication complexity $O(\log n)$ elements of a bilinear module. The prover performs $O(n)$ operations over the ring and modules of the leveled bilinear-module system; and the verifier performs $O(\log^2 n)$ such operations. (Indeed, in the i-th sumcheck argument the prover performs $O(n/2^i)$ operations and the verifier performs $O(\log n - i)$ operations.)

Completeness of the protocol is straightforward, since the i-th iteration reduces a true statement about a polynomial evaluation over the i-th level into a true statement about a polynomial evaluation over the $(i + 1)$-th level, using the embedding map up_i. The verifier accepts because each iteration is a sumcheck argument for a valid polynomial evaluation. In contrast, establishing soundness requires more care, as we now explain.

Soundness. The protocol consists of $\log n$ sumcheck arguments, so a starting point for arguing soundness is to follow the approach in [BCS21]. There, a valid witness is extracted from an extraction tree (a collection of accepting transcripts with a special tree-like structure). For instance, in the case of polynomial commitments as in Protocol 2, the extraction tree is a ternary tree of depth $\log n$. An extraction tree can be obtained, from a suitable malicious prover, in time exponential in its depth (e.g. see the forking lemma in [ACK21, Lemma 5]). While this technique works in a single iteration of Protocol 3 to prove knowledge soundness, it fails when applied in the final delegation protocol which consists of $\log n$ iterations. This is because now we would need an extraction tree of depth $\log^2 n$, and producing such a tree takes quasi-polynomial time.

An alternative approach is to start from the knowledge soundness of each iteration of Protocol 3, which is based on an extraction tree of depth only $\log n$. Informally, the soundness of the final delegation protocol then follows by a union bound on the $\log n$ iterations. This approach is used, e.g., to establish the soundness of the $O(\log^2 n)$-round version of [Lee21] presented in [Tha22]. However, in our case, which also captures the lattice setting, this has a negative impact in the parameters.

For example, in the lattice setting, it is only known how to prove knowledge soundness of Protocol 3 for a relaxed statement [BCS21]. More precisely, if the verifier accepts in Protocol 3, then we can extract a *relaxed opening* $\underline{\mathsf{m}} \in M_{\mathrm{L},1}^n$ to C such that $c \cdot \mathsf{C} = \langle \underline{\mathsf{m}}, \underline{\mathsf{G}} \rangle$ and $p_{\underline{\mathsf{m}}}(\underline{z}) = u$, where c is called the *slackness*. Then, establishing soundness by simply applying the knowledge soundness property of Task 6 recursively ℓ times causes the slackness to accumulate at each extraction step. This approach can only prove that the final delegation protocol has slackness exponential in $\log n$.

We avoid the accumulation of slackness by leveraging the fact that the statement to be proved is a deterministic computation: if the prover does not send a correct evaluation of the key polynomial at the end of each iteration, then the verifier rejects (with some good probability). There is no witness to extract, since the commitment keys are part of the public parameters. In the security proof we can check whether the prover sends an incorrect evaluation in each iteration of Protocol 3. If any of the evaluations is incorrect, then we extract a message that breaks the binding property of the commitment of this iteration. The i-th iteration of Protocol 3 has soundness error $O(\frac{\log n - i}{|\mathcal{C}|})$; hence, the soundness error of the entire protocol is $O(\frac{\log n^2}{|\mathcal{C}|})$. The final slackness remains c.

From relaxed to exact openings. Relaxed openings prove approximate statements about polynomial evaluations. This is a problem when we wish to reason about exact satisfiability of algebraic relations, such as R1CS. We modify the polynomial commitment scheme to allow us to divide out the slackness, and hence to extract exact openings. Specifically, we consider $M_{\mathrm{L},1}$ to be a ring and I an ideal of $M_{\mathrm{L},1}$ in which multiplication by slackness c is invertible. Then, intuitively, an opening of a commitment $c \cdot \mathsf{C}$ to message $\underline{\mathsf{m}} \in M_{\mathrm{L},1}^n$ can be viewed as an opening of C to $c^{-1}\underline{\mathsf{m}} \in (M_{\mathrm{L},1}/I)^n$. The message space for the modified commitment scheme is $M_{\mathrm{L},1}/I$. To commit to a polynomial with coefficients in $M_{\mathrm{L},1}/I$, we first lift them to elements in $M_{\mathrm{L},1}$ and then apply the original, unmodified commitment scheme. Specifically, our lattice-based instantiation of the leveled modules and rings leads to a polynomial commitment scheme over a ring R/pR.

2.6 Polynomial IOP for product rings

As described in Section 2.1, our succinct argument is obtained by combining the polynomial commitment scheme described in Section 2.5 and a polynomial IOP (PIOP). In a PIOP, the prover can send polynomials to the verifier as oracle messages, and the verifier's queries request evaluations of these polynomials.

While there are PIOPs that work over finite fields $\mathbb{F}$, to prove Theorem 2 we need a PIOP that works over rings satisfying $R_\bullet \simeq \mathbb{F}^k$. This suffices to prove Theorem 1 as a special case of Theorem 2 because the cyclotomic rings that arise from the lattice instantiation can be expressed as product rings using facts from algebraic number theory.[2]

[2]In more detail, consider a cyclotomic ring of the form $R := \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$ where $\Phi_d(X)$ is the d-th cyclotomic polynomial.

PIOPs over product rings. We obtain a holographic PIOP for R1CS over product rings $R_\bullet \simeq \mathbb{F}^k$ by using k times "in parallel" an existing PIOP construction over $\mathbb{F}$, as we now explain. First, we apply the isomorphism between $R_\bullet$ and $\mathbb{F}^k$ to an R1CS instance defined over $R_\bullet$, producing k R1CS instances defined over $\mathbb{F}$. Observe that the non-zero entries in each of the k R1CS instances over $\mathbb{F}$ are a subset of the non-zero entries in the instance over $R_\bullet$. Second, we use the holographic PIOP with succinct verification for R1CS instances over $\mathbb{F}$ from prior work [BCG20]. More precisely, we run this PIOP for the k R1CS instances over $\mathbb{F}$ using the same random verifier challenges (which are sampled from $\mathbb{F}$). This gives a PIOP with similar complexity parameters defined over $R_\bullet$ by mapping all of the prover and verifier messages back into $R_\bullet$.

This approach works because the PIOP in [BCG20] has the following special property: the indexer, prover, and verifier can be modeled as arithmetic circuits which have hard-coded the positions of non-zero entries in the R1CS instance[3]. Since the set of non-zero entries in the R1CS instance over $R_\bullet$ is a superset of the non-zero entries in the k R1CS instances over $\mathbb{F}$, the arithmetic circuits for the indexer, prover, and verifier are the same for the k instances over $\mathbb{F}$. Thus, a PIOP for R1CS over $\mathbb{F}$ can be converted into a PIOP over $R_\bullet$ with the same proof size and computational complexity as the original PIOP, but measured as elements and operations over $R_\bullet$.

In sum, we obtain a ring-based PIOP with linear prover time and logarithmic verifier time.

Lemma 1 (informal)**.** *For every ring $R_\bullet$ such that $R_\bullet \simeq \mathbb{F}^k$, there is a holographic polynomial IOP for R1CS over the ring $R_\bullet$ with instances of size N with M non-zero entries, with the following properties:*

- *the round complexity is $O(\log(M+N))$;*
- *the proof length is $O(M+N)$ elements in $R_\bullet$;*
- *the query complexity is $O(1)$;*
- *the communication complexity is $O(\log(M+N))$ messages in $R_\bullet$;*
- *the indexer uses $O(M)$ operations in $R_\bullet$;*
- *the prover uses $O(N+M)$ operations in $R_\bullet$;*
- *the verifier uses $O(\log M)$ operations in $R_\bullet$.*

Here, "proof length" refers to the total number of elements of $R_\bullet$ in oracle messages, while "communication complexity" refers to the total number of (non-oracle) message elements received by the verifier.

2.7 Final protocol: combining polynomial commitments and PIOP

To obtain Theorem 2, we combine the polynomial commitment scheme described in Section 2.5 and the PIOP over product rings of Section 2.6. Then, Theorem 1 follows as a special case by using the lattice-based instantiation of a leveled bilinear module.

Protocol 5: succinct interactive argument for R1CS over $\underline{\mathcal{M}}$

Setup. On input $N \in \mathbb{N}$, the setup algorithm runs the setup algorithm for the polynomial commitment scheme to generate public parameters for committing to messages of length N. As part of this algorithm, the setup algorithm samples a levelled bilinear module with $\underline{\mathcal{M}}$, containing the description of a ring $M_{\mathrm{L},1}$, an ideal I_1, and a module $M_{\mathrm{T},\ell}$, where $\ell = \log(N)$.

Indexer. On input an R1CS instance of size N with M non-zero entries defined over the ring

The polynomial $\Phi_d(X)$ modulo a prime p with $\gcd(p,d) = 1$ factors into irreducible polynomials of the same degree t for some $t \in \mathbb{N}$ (e.g., from [Con13, Theorem 5.3]). This means that R/pR is isomorphic to $k := \phi(d)/t$ copies of $\mathbb{F}_{p^t}$.

[3]This is despite the fact that the PIOP construction in full generality sometimes uses non-algebraic operations such as linear scans.

$R_\bullet = M_{\mathrm{L},1}/I_1 \simeq \mathbb{F}^k$, the indexer algorithm runs the indexer algorithm for the PIOP for $R_\bullet$ of Section 2.6, producing polynomial oracle messages defined over $R_\bullet$. Then the indexer runs the indexer of the polynomial commitment scheme of Section 2.5, and computes commitments to each of the polynomials. The indexer computes a proving key ipk consisting of the polynomials, their commitments, and the proving key for the polynomial commitment scheme. The indexer computes a verification key ivk consisting of the commitments and the verification key for the polynomial commitment scheme. Finally, the indexer outputs ipk and ivk.

Prover and verifier. The prover receives ipk, while the verifier receives ivk. The prover and verifier run the prover and verifier algorithms for the PIOP of Section 2.6, forwarding messages between the PIOP prover and verifier. Whenever the PIOP prover produces a polynomial oracle message over $R_\bullet$, the prover commits to it using the polynomial commitment scheme and sends the result to the verifier. Whenever the PIOP verifier makes a polynomial evaluation query, the verifier forwards it to the prover, who evaluates the polynomial, and sends the evaluation back to the verifier. The prover and verifier then use the polynomial commitment scheme to prove that the evaluation is consistent with the correct committed polynomial. The verifier accepts if all evaluations are consistent, and the PIOP verifier acccepts.

The verifier must perform $O(\log M)$ operations over $R_\bullet$ as part of the PIOP, and $O(\log^2(M+N))$ operations over $M_{\mathrm{T},\ell}$ to use the polynomial commitment scheme to verify each of the $O(1)$ PIOP query responses. The communication complexity of the argument is dominated by the $O(\log^2(M+N))$ elements of $M_{\mathrm{T},\ell}$ sent when using the polynomial commitment scheme. This yields a succinct argument with efficient verification for NP over a leveled bilinear-module system.

3 Preliminaries

3.1 Rings and modules

A ring R is a mathematical structure that generalizes a field: R is equipped with addition and multiplication operations, but, unlike in a field, multiplicative inverses need not exist. We use commutative rings, where the multiplication operation commutes. The multiplicative subgroup of R is denoted $R^{\times}$. A module M over a ring R extends the notion of vector space over a field, where the scalars are elements of a ring.

Norms. We use rings and modules equipped with norms. The definitions below are slightly different than the ones in standard algebra textbooks due to the use of expansion factors.

Definition 3.1. *Let R be a ring. A* **norm for** *R is a map $\|\cdot\|_R\colon R \to \mathbb{R}_{\geq 0}$ that satisfies the following properties: (i) $\|0\|_R = 0$ and $\|1\|_R = 1$; (ii) for every $a \in R$, $\|a\|_R = \|-a\|_R$; (iii) for every $a, b \in R$, $\|a+b\|_R \leq \|a\|_R + \|b\|_R$; (iv) there exists a constant "augmentation factor" $\gamma_R \in \mathbb{R}_{>0}$ such that, for every $a, b \in R$, $\|ab\|_R \leq \gamma_R \|a\|_R \|b\|_R$.*

Definition 3.2. *Let R be a ring with norm $\|\cdot\|_R$, and let M be an R-module. A* **norm for** *M is a map $\|\cdot\|_M\colon R \to \mathbb{R}_{\geq 0}$ that satisfies the following properties: (i) $\|0\|_M = 0$; (ii) for every $a \in M$, $\|a\|_M = \|-a\|_M$; (iii) for every $a, b \in M$, $\|a+b\|_M \leq \|a\|_M + \|b\|_M$; (iv) there exists a constant "augmentation factor" $\gamma_M \in \mathbb{R}_{>0}$ such that, for every $a \in R$ and $b \in M$, $\|ab\|_M \leq \gamma_M \|a\|_R \|b\|_M$.*

Remark 3.3. To simplify notation, while multiplication of elements of M and R may cause norm increases by different factors γ_R and γ_M, we only use the notation γ_R to represent the maximum of these quantities.

Definition 3.4. *For a ring R with norm $\|\cdot\|_R$, $R(B) := \{r \in R : \|r\|_R \leq B\}$ is the set of ring elements with norm at most B; and similarly for a module M and set $M(B)$. For a set $\mathcal{C} \subseteq R$, $\|\mathcal{C}\|_R := \max_{x \in \mathcal{C}} \|x\|_R$.*

For a normed module M, the norm of a vector $\underline{v} \in M^n$ is $\|\underline{v}\|_M := \max_{i \in [n]} \|v_i\|_M$ (the maximum of the norms of all entries of $\underline{v}$).

Polynomials over modules. We define the multilinear polynomial associated with a vector over a module. We will use these polynomials to prove properties of vectors using the sumcheck protocol over modules.

Definition 3.5. *Let R be a ring and M an R-module. For $n \in \mathbb{N}$ a power of 2, set $\ell := \log n$ and let $\underline{v} \in M^n$ be a vector whose entries we index via binary strings $(i_1, \ldots, i_\ell) \in \{0,1\}^\ell$. The ℓ-variate polynomial $p_{\underline{v}} \in M[X_1, \ldots, X_\ell]$ is defined as follows:*

$$p_{\underline{v}}(X_1, \ldots, X_\ell) := \sum_{i_1, \ldots, i_\ell \in \{0,1\}} v_{i_1, \ldots, i_\ell} X_1^{i_1} \cdots X_\ell^{i_\ell} \ .$$

Lemma 3.6 ([BCS21, Lemma 3.6])**.** *Let $\mathcal{H}$ be a cyclic subgroup (of finite order) of the multiplicative group of a ring R, such that $1 - h$ is not a zero-divisor for every $h \in \mathcal{H} \setminus \{1\}$. Let M be an R-module and let $P(X_1, \ldots, X_\ell) \in M[X_1, \ldots, X_\ell]$ be a polynomial. If we denote by $P_{i_1, \ldots, i_\ell} \in M$ the coefficient of $X_1^{i_1} \cdots X_\ell^{i_\ell}$ in the polynomial $P(X_1, \ldots, X_\ell)$, then*

$$\sum_{\underline{\omega} \in \mathcal{H}^\ell} P(\underline{\omega}) = \left(\sum_{\underline{i} \equiv \underline{0} \bmod |\mathcal{H}|} P_{\underline{i}} \right) \cdot |\mathcal{H}|^\ell \ .$$

3.2 Interactive arguments with preprocessing

Definition 3.7. *An* **indexed relation** $\mathcal{R}(\mathsf{pp})$ *parametrized by* pp *is a set of tuples* $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ *where* $\mathbb{i}$ *is the index,* $\mathbb{x}$ *the instance, and* $\mathbb{w}$ *the witness. The corresponding* **indexed language** $\mathcal{L}_{\mathcal{R}}(\mathsf{pp})$ *is the set of pairs* $(\mathbb{i}, \mathbb{x})$ *for which there exists a witness* $\mathbb{w}$ *such that* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}(\mathsf{pp})$.

$\mathsf{ARG} = (\mathbf{G}, \mathbf{I}, \mathbf{P}, \mathbf{V})$ is an *interactive argument with preprocessing* for a parametrized indexed relation $\mathcal{R}(\mathsf{pp})$ if it satisfies the following completeness and soundness properties.

- **Completeness.** For all $\lambda, N \in \mathbb{N}$, and all adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}(\mathsf{pp}) \text{ or} & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda, N) \\ b = 1 & (\mathbb{i}, \mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ & (\mathsf{ipk}, \mathsf{ivk}) := \mathbf{I}(\mathsf{pp}, \mathbb{i}) \\ & b \leftarrow \langle \mathbf{P}(\mathsf{ipk}, \mathbb{x}, \mathbb{w}), \mathbf{V}(\mathsf{ivk}, \mathbb{x}) \rangle \end{array}\right] = 1 \ ,$$

  where b is the verifier's output at the end of this interaction.

- **Soundness.** ARG has soundness error $\epsilon \colon \mathbb{N} \times \mathbb{N} \to [0, 1)$ if for all $\lambda, N \in \mathbb{N}$, and all polynomial-size adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}(\mathsf{pp}) \text{ and} & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda, N) \\ b = 1 & (\mathbb{i}, \mathbb{x}, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ & (\mathsf{ipk}, \mathsf{ivk}) := \mathbf{I}(\mathsf{pp}, \mathbb{i}) \\ & b \leftarrow \langle \mathcal{A}(\mathsf{aux}), \mathbf{V}(\mathsf{ivk}, \mathbb{x}) \rangle \end{array}\right] \leq \epsilon(\lambda, N) \ ,$$

  where b is the verifier's output at the end of this interaction.

We also consider the stronger than soundness property of witness-extended emulation.

- **Witness-extended emulation.** ARG has witness-extended emulation with knowledge error $\kappa \colon \mathbb{N} \times \mathbb{N} \to [0, 1)$ if there exists an expected polynomial-time algorithm $\mathcal{E}$ such that for all $\lambda, N \in \mathbb{N}$, and all polynomial-size adversaries $\mathcal{A}$,

$$\left| \Pr\left[\begin{array}{c|r} \mathcal{A}(\mathsf{aux}, \mathrm{tr}) = 1 & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda, N) \\ & (\mathbb{i}, \mathbb{x}, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ & (\mathsf{ipk}, \mathsf{ivk}) := \mathbf{I}(\mathsf{pp}, \mathbb{i}) \\ & b \overset{\mathrm{tr}}{\leftarrow} \langle \mathcal{A}(\mathsf{aux}), \mathbf{V}(\mathsf{ivk}, \mathbb{x}) \rangle \end{array}\right] - \Pr\left[\begin{array}{c|r} \mathcal{A}(\mathsf{aux}, \mathrm{tr}) = 1 & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda, N) \\ \text{and} & (\mathbb{i}, \mathbb{x}, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \text{if } \mathrm{tr} \text{ is accepting then } (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}(\mathsf{pp}) & (\mathrm{tr}, \mathbb{w}) \leftarrow \mathcal{E}^{\mathcal{A}(\mathsf{aux})}(\mathsf{pp}, \mathbb{i}, \mathbb{x}) \end{array}\right] \right| \leq \kappa(\lambda, N) \ .$$

  Above, tr is the transcript of the interaction between $\mathbf{P}$ and $\mathbf{V}$, b is the verifier's output at the end of this interaction, and $\mathcal{E}$ has oracle access to (the next-message functions of) $\mathcal{A}(\mathsf{aux})$.

3.3 Holographic polynomial IOPs

A holographic public-coin polynomial IOP over a ring family R for an indexed relation $\mathcal{R}$ is specified by a tuple

$$\mathsf{IOP} = (\mathsf{k}, \mathsf{s}, \mathsf{d}, \mathcal{I}, \mathcal{P}, \mathcal{V})$$

where $\mathsf{k}, \mathsf{s}, \mathsf{d} \colon \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions and $\mathcal{I}, \mathcal{P}, \mathcal{V}$ are three algorithms known as the *indexer*, *prover*, and *verifier*. The parameter k specifies the number of interaction rounds, s specifies the number of polynomials in each round, and d specifies degree bounds on these polynomials.

In the offline phase ("0-th round"), the indexer $\mathcal{I}$ receives as input a ring $R \in \mathsf{R}$ and an index $\mathbb{i}$ for $\mathcal{R}$, and outputs $\mathsf{s}(0)$ polynomials $p_1^{(0)}, \dots, p_{\mathsf{s}(0)}^{(0)} \in R[X]$ of degrees at most $\mathsf{d}(|\mathbb{i}|, 0, 1), \dots, \mathsf{d}(|\mathbb{i}|, 0, \mathsf{s}(0))$ respectively. Note that the offline phase does not depend on any particular instance or witness, and merely considers the task of encoding the given index $\mathbb{i}$.

In the online phase, given an instance $\mathbb{x}$ and witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the prover $\mathcal{P}$ receives $(R, \mathbb{i}, \mathbb{x}, \mathbb{w})$ and the verifier $\mathcal{V}$ receives $(R, \mathbb{x})$ and oracle access to the polynomials output by $\mathcal{I}(R, \mathbb{i})$. The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ interact over $\mathsf{k} = \mathsf{k}(|\mathbb{i}|)$ rounds. For $j \in [\mathsf{k}]$, in the j-th round of interaction, the verifier $\mathcal{V}$ sends a message $\rho_j \in R$ to the prover $\mathcal{P}$; then the prover $\mathcal{P}$ replies with $\mathsf{s}(j)$ oracle polynomials $p_1^{(j)}, \dots, p_{\mathsf{s}(j)}^{(j)} \in R[X]$. The verifier may query any of the polynomials it has received any number of times. A query consists of a location $z \in R$ for an oracle $p_i^{(j)}$, and its corresponding answer is $p_i^{(j)}(z) \in M$. After the interaction, the verifier accepts or rejects.

The function d determines which provers to consider for the completeness and soundness properties of the proof system. In more detail, we say that a (possibly malicious) prover $\tilde{\mathcal{P}}$ is **admissible** for IOP if, on every interaction with the verifier $\mathcal{V}$, it holds that for every round $j \in [\mathsf{k}]$ and oracle index $i \in [\mathsf{s}(j)]$ we have $\deg(p_i^{(j)}) \le \mathsf{d}(|\mathbb{i}|, j, i)$. The honest prover $\mathcal{P}$ is required to be admissible under this definition.

Let $\epsilon \colon \{0,1\}^* \to [0,1)$. We say that IOP has perfect completeness and soundness error ϵ if the following holds.

- **Completeness.** For every ring $R \in \mathsf{R}$ and index-instance-witness tuple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the probability that $\mathcal{P}(R, \ \mathbb{i}, \mathbb{x}, \mathbb{w})$ convinces $\mathcal{V}^{\mathcal{I}(R,\mathbb{i})}(R, \mathbb{x})$ to accept in the interactive oracle protocol is 1.
- **Soundness.** For every ring $R \in \mathsf{R}$, index-instance pair $(\mathbb{i}, \mathbb{x}) \notin \mathcal{L}(\mathcal{R})$, and admissible prover $\tilde{\mathcal{P}}$, the probability that $\tilde{\mathcal{P}}$ convinces $\mathcal{V}^{\mathcal{I}(R,\mathbb{i})}(R, \mathbb{x})$ to accept in the interactive oracle protocol is at most $\epsilon(R, \mathbb{x})$.

The *proof length* l is the sum of all degree bounds in the offline and online phases, namely $\mathsf{l}(|\mathbb{i}|) := \sum_{j=0}^{\mathsf{k}(|\mathbb{i}|)} \sum_{i=1}^{\mathsf{s}(j)} \mathsf{d}(|\mathbb{i}|, j, i)$.

The *query complexity* q is the total number of queries made by the verifier to the polynomials. This includes queries to the polynomials output by the indexer and those sent by the prover.

3.4 Commitments

A (non-interactive) *commitment scheme* is a tuple of polynomial-time probabilistic algorithms $\mathsf{CM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Open})$ with the following syntax.

- $\mathsf{CM.Setup}(1^\lambda, n) \to \mathsf{pp}_{\mathsf{CM}}$: Sample public parameters given a security parameter and a message length. The public parameters $\mathsf{pp}_{\mathsf{CM}}$ determine a commitment space $\mathbb{C}_{\mathsf{pp}_{\mathsf{CM}}}$, key space $\mathbb{K}_{\mathsf{pp}_{\mathsf{CM}}}$, message space $\mathbb{M}_{\mathsf{pp}_{\mathsf{CM}}}$, and opening space $\mathbb{O}_{\mathsf{pp}_{\mathsf{CM}}}$.
- $\mathsf{CM.KeyGen}(\mathsf{pp}_{\mathsf{CM}}) \to \mathsf{ck}$: Sample a commitment key.
- $\mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m}) \to \mathsf{cm}$: Use the commitment key ck to compute a commitment $\mathsf{cm} \in \mathbb{C}_{\mathsf{pp}_{\mathsf{CM}}}$ to $\mathsf{m} \in \mathbb{M}_{\mathsf{pp}_{\mathsf{CM}}}$. If $\mathsf{ck} \notin \mathbb{K}_{\mathsf{pp}_{\mathsf{CM}}}$ or $\mathsf{m} \notin \mathbb{M}_{\mathsf{pp}_{\mathsf{CM}}}$, it outputs $\perp$.
- $\mathsf{CM.Open}(\mathsf{ck}, \mathsf{m}, \mathsf{cm}, o, c) \to b \in \{0,1\}$: Checks that $\mathsf{cm} \in \mathbb{C}_{\mathsf{pp}_{\mathsf{CM}}}$ is a commitment to the message $\mathsf{m} \in \mathbb{M}_{\mathsf{pp}_{\mathsf{CM}}}$ with opening value and slackness $(o, c) \in \mathbb{O}_{\mathsf{pp}_{\mathsf{CM}}}$, relative to the commitment key ck.

We require CM to satisfy the following completeness and binding properties. Though commitment schemes typically also satisfy a *hiding* property, we omit it for simplicity as it is not required for our constructions.

Definition 3.8. CM *is* **complete** *if for all* $\lambda, n \in \mathbb{N}$*, and all adversaries* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{CM.Open}\left(\mathsf{ck}, \mathsf{m}, \mathsf{cm}, \bot, \bot\right) = 1 \;\middle|\; \begin{array}{c} \mathsf{pp}_{\mathsf{CM}} \leftarrow \mathsf{CM.Setup}(1^\lambda, n) \\ \mathsf{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp}_{\mathsf{CM}}) \\ (\mathsf{m} \in \mathbb{M}_{\mathsf{ck}}) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck}) \\ \mathsf{cm} \leftarrow \mathsf{CM.Commit}\left(\mathsf{ck}, \mathsf{m}\right) \end{array}\right] = 1 \ .$$

Definition 3.9. CM *is (computationally)* **binding** *with error* $\epsilon \colon \mathbb{N} \times \mathbb{N} \to [0,1)$ *if for all* $\lambda, n \in \mathbb{N}$*, and all polynomial-size adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c} \mathsf{m}_0 \neq \mathsf{m}_1 \\ \mathsf{CM.Open}\left(\mathsf{ck}, \mathsf{m}_0, \mathsf{cm}, o_0, c\right) = 1 \\ \mathsf{CM.Open}\left(\mathsf{ck}, \mathsf{m}_1, \mathsf{cm}, o_1, c\right) = 1 \end{array} \;\middle|\; \begin{array}{r} \mathsf{pp}_{\mathsf{CM}} \leftarrow \mathsf{CM.Setup}(1^\lambda, n) \\ \mathsf{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp}_{\mathsf{CM}}) \\ (\mathsf{cm}, \mathsf{m}_0, \mathsf{m}_1, o_0, o_1, c) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck}) \end{array}\right] \leq \epsilon(\lambda, N) \ .$$

3.4.1 Polynomial commitment schemes

A commitment scheme is called a *polynomial commitment scheme* over a module family M if it satisfies completeness and binding, and there is an interactive argument with preprocessing, $\mathsf{PC\text{-}Eval} = (\mathbf{G}, \mathbf{I}, \mathbf{P}, \mathbf{V})$ for the indexed relation $\mathcal{R}_{\mathrm{PC}}$:

$$\mathcal{R}_{\mathrm{PC}}(\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck}, n) := \left\{ (\bot, (\mathsf{cm}, z, v), (P, o, c)) \;\middle|\; \begin{array}{c} \mathsf{pp}_{\mathsf{CM}} \in \mathsf{CM.Setup}(1^\lambda, n), \\ \text{where } \mathbb{M}_{\mathsf{pp}_{\mathsf{CM}}} = M^{\leq n}[X_1, \ldots, X_\ell], \\ \mathsf{ck} \in \mathsf{CM.KeyGen}(\mathsf{pp}_{\mathsf{CM}}) \\ \mathsf{cm} \in \mathbb{C}_{\mathsf{pp}_{\mathsf{CM}}}, z \in R, v \in M \\ P \in M^{\leq n}[X_1, \ldots, X_\ell], (o, c) \in \mathbb{O}_{\mathsf{pp}_{\mathsf{CM}}}, \\ P(z) = v, \\ \mathsf{CM.Open}\left(\mathsf{ck}, P, \mathsf{cm}, o, c\right) = 1 \end{array} \right\} .$$

As with commitment schemes, polynomial commitment schemes typically satisfy a hiding property, which we omit as it will not be used.

3.5 Sumcheck arguments

Sumcheck arguments [BCS21] are interactive arguments for commitment openings over bilinear modules. In this paper, we use sumcheck arguments for proving evaluations of polynomial commitments. We summarize some useful notions related to sumcheck arguments from [BCS21] and restate results on sumcheck arguments for construction of polynomial commitments.

Definition 3.10. *Let* R *be a ring with norm* $\|\cdot\|_R$ *and* $\mathcal{C} \subseteq R$ *a set. Let* $V_{c_1,\ldots,c_K}$ *be the Vandermonde matrix with respect to distinct* $c_1, \ldots, c_K \in \mathcal{C}$*:*

$$V_{c_1,\ldots,c_K} := \begin{bmatrix} 1 & c_1 & \cdots & c_1^{K-1} \\ 1 & c_2 & \cdots & c_2^{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & c_K & \cdots & c_K^{K-1} \end{bmatrix}$$

and let $A_{c_1,\dots,c_K}$ be the adjugate of $V_{c_1,\dots,c_K}$ (which satisfies $A_{c_1,\dots,c_K} \cdot V_{c_1,\dots,c_K} = \det(V_{c_1,\dots,c_K}) \cdot I_K$).

*For $K \in \mathbb{N}$, we define the K-**th inversion constant** associated to $\mathcal{C}$ to be*

$$\iota(\mathcal{C}, K) := \max_{c_1,\dots,c_K \in \mathcal{C}} \max_{i,j \in [K]} \|A_{c_1,\dots,c_K}[i,j]\|_R \enspace .$$

Definition 3.11. *A **bilinear module** is a tuple $\mathcal{M} = (R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e)$ where R is a ring, $M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}$ are R-modules, and $e \colon M_{\mathrm{L}} \times M_{\mathrm{R}} \to M_{\mathrm{T}}$ is an R-bilinear map; moreover, R and M_{L} are equipped with norms $\|\cdot\|_R$ and $\|\cdot\|_{M_{\mathrm{L}}}$. We use arithmetic notation as a shorthand for e: for $a \in M_{\mathrm{L}}$ and $\mathsf{G} \in M_{\mathrm{R}}$, "$a \cdot \mathsf{G}$" denotes $e(a, \mathsf{G}) \in M_{\mathrm{T}}$; similarly, for $\underline{a} \in M_{\mathrm{L}}^n$ and $\underline{\mathsf{G}} \in M_{\mathrm{R}}^n$, "$\langle \underline{a}, \underline{\mathsf{G}} \rangle$" denotes $\sum_{i \in [n]} e(a_i, \mathsf{G}_i) \in M_{\mathrm{T}}$.*

Definition 3.12. *Let R be a ring and M an R-module. For $\xi \in R$ and $D \in \mathbb{N}$ and $\mathcal{C} \subseteq R$ a set, we say that $(\mathcal{C}, \xi, D)$ are **pseudoinverse parameters** for (R, M) if for every $a \in R$, $m, m^* \in M$, and distinct $c_1, c_2 \in \mathcal{C}$ it holds that if $(c_1 - c_2)m = a \cdot m^*$ then there exists $r \in R$ such that $\xi \cdot m = r \cdot m^*$ and $\|r\|_R \le D\|a\|_R$.*

We define bilinear-module generators similarly to [BCS21, Definition 5.4].

Definition 3.13. *A **bilinear-module generator** is a tuple $\mathsf{BM} = (\mathsf{Setup}, \mathsf{KeyGen})$ with the following syntax:*
- $\mathsf{BM.Setup}$, *given 1^λ and $n \in \mathbb{N}$, outputs a bilinear module $\mathcal{M}$ and auxiliary string* aux;
- $\mathsf{BM.KeyGen}$, *given $(\mathcal{M}, n, \mathsf{aux})$ where $n \le N$, outputs a vector in M_{R}^n.*

We assume that the parameters λ and n are part of aux.

The bilinear-module generators that we consider output auxiliary strings that contain several pieces of information: $\mathsf{aux} = (B_{\mathsf{BRA}}, \mathcal{C}, \xi, D, B_{\mathsf{c}})$ where $B_{\mathsf{BRA}} \in \mathbb{R}$, $\mathcal{C} \subseteq R$, $\xi \in R$, $D \in \mathbb{R}$, and $B_{\mathsf{c}} \in \mathbb{R}$ with $B_{\mathsf{c}} \le B_{\mathsf{BRA}}$.

We consider bilinear module generators with specific properties. In particular, we define secure and quotient-friendly bilinear module generators. Our security definition differs from [BCS21, Definition 5.6] as we remove the definition of the hiding property and associated parameters, such as the integer h which lengthens the commitment key to allow space for commitment randomness.

Definition 3.14. *A bilinear-module generator* BM *is **secure** if it satisfies the following properties.*

- BM *satisfies the **bilinear relation assumption (BRA)**: there exists $\epsilon \colon \mathbb{N} \times \mathbb{N} \to [0,1)$ such that for all $\lambda \in \mathbb{N}$, $n \ge 2$, algorithm* Check, *and all polynomial-size adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c|r} \mathsf{Check}(\mathcal{M}, \mathsf{aux}) = 1 & \\ \underline{a} \in M_{\mathrm{L}}^n(B_{\mathsf{BRA}}) & (\mathcal{M}, \mathsf{aux}) \leftarrow \mathsf{BM.Setup}(1^\lambda, n) \\ \underline{a} \ne \{0^n\} & \underline{\mathsf{G}} \leftarrow \mathsf{BM.KeyGen}(\mathcal{M}, \mathsf{aux}) \\ \langle \underline{a}, \underline{\mathsf{G}} \rangle = 0 & \underline{a} \leftarrow \mathcal{A}(\mathcal{M}, \mathsf{aux}, \underline{\mathsf{G}}) \end{array}\right] \le \epsilon(\lambda, N) \enspace ;$$

- $(\mathcal{C}, \xi, D)$ *in* aux *are pseudoinverse parameters for (R, M_{T}) output by* $\mathsf{BM.Setup}$ *(Definition 3.12).*

The "Check" algorithm in Definition 3.14 is used to determine whether the parameters of $\mathcal{M}$ and aux are suitable for use as part of larger algorithms which may introduce additional constraints.[4]

[4]For example, suppose that the output of BM is to be used as part of a succinct argument for R1CS instances of size N with M non-zero entries over $\mathbb{Z}_p$. Then, the parameters in the auxiliary string aux may need to satisfy various inequalities in terms of N, M and p. However, providing N, M and p to the $\mathsf{BM.Setup}$ and $\mathsf{BM.KeyGen}$ algorithms specializes the bilinear module generator to this particular R1CS application, and makes it difficult to use the same generator in other proof systems which may enforce other constraints. To avoid this, the Check algorithm verifies that the constraints are satisfied after parameter generation. It is relatively simple to combine two Check algorithms into one.

Definition 3.15. BM *is* **quotient-friendly** *if* M_{L} *output by* $\mathsf{BM.Setup}$ *contains an* R*-submodule* $I \subseteq M_{\mathrm{L}}$ *such that* $(\mathcal{C}, \xi, D)$ *in* aux *output by* $\mathsf{BM.Setup}$ *are pseudoinverse parameters for* (R, M_{T})*, the multiplication by* ξ *is invertible in* M_{L}/I *and each element of* M_{L}/I *has at least one representative in* $M_{\mathrm{L}}(B_{\mathsf{BRA}})$*.*

We restate the construction of sumcheck arguments, and their properties, to account for small changes in this paper. Sumcheck arguments are arguments for commitment openings when the commitment scheme has a special structure which we call *sumcheck-friendly*.

Definition 3.16. *A commitment scheme* $\mathsf{CM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Open})$ *is* **sumcheck-friendly** *if for every security parameter* $\lambda \in \mathbb{N}$*, message length* $n \in \mathbb{N}$*, and public parameters* $\mathsf{pp}_{\mathsf{CM}} \in \mathsf{CM.Setup}(1^{\lambda}, n)$*,* $\mathbb{O}_{\mathsf{ck}} = \mathbb{S}_{\mathsf{ck}}$ *for a* slackness space $\mathbb{S}_{\mathsf{ck}}$*, and there exist a ring* R*, domain* $\mathcal{H} \subseteq R$*, challenge set* $\mathcal{C} \subseteq R$*, number of variables* $\ell \in \mathbb{N}$*,* R*-modules* $\mathbb{M}, \mathbb{K}, \mathbb{C}$ *with* $\mathbb{M}$ *having a norm, and efficient functions* $f_{\mathsf{CM}}, \phi_{\mathrm{sc}}, \alpha_{\mathrm{sc}}$ *such that for every commitment key* $\mathsf{ck} \in \mathsf{CM.KeyGen}(\mathsf{pp}_{\mathsf{CM}})$*, message* $\mathsf{m} \in \mathbb{M}_{\mathsf{ck}}$*, and slackness* $c \in \mathbb{S}_{\mathsf{ck}}$*:*

- $\mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m}) = \sum_{\underline{\omega} \in \mathcal{H}^{\ell}} f_{\mathsf{CM}}(P_{\mathsf{m}}(\underline{\omega}), P_{\mathsf{ck}}(\underline{\omega}), 1)$*; and*
- $\phi_{\mathrm{sc}}\big(\mathsf{cm}, \sum_{\underline{\omega} \in \mathcal{H}^{\ell}} f_{\mathsf{CM}}(P_{\mathsf{m}}(\underline{\omega}), P_{\mathsf{ck}}(\underline{\omega}), c), c\big) = 1$ *if and only if* $\mathsf{CM.Open}(\mathsf{ck}, \mathsf{m}, \mathsf{cm}, c) = 1$*;*
- *when* $c = 1$*,* ϕ_{sc} *is simply an equality check on its first two inputs;*
- *for every* $i \in \{0, 1, \dots, \ell\}$*,* $p \in \mathbb{M}[X_{i+1}, \dots, X_{\ell}]$*, and* $(r_1, \dots, r_i) \in \mathcal{C}^i$*,* $\alpha_{\mathrm{sc}}(\mathsf{ck}, p, r_1, \dots, r_i) = 1$ *if and only if there exists a message* $\mathsf{m} \in \mathbb{M}_{\mathsf{ck}}$ *such that* $p(X_{i+1}, \dots, X_{\ell}) = P_{\mathsf{m}}(r_1, \dots, r_i, X_{i+1}, \dots, X_{\ell})$*.*

Here:

- $P_{\mathsf{m}}(X_1, \dots, X_{\ell})$ *is a polynomial over* $\mathbb{M}$ *that can be efficiently obtained from the message* m *(and, conversely,* m *can be efficiently obtained from* P_{m}*);*
- $P_{\mathsf{ck}}(X_1, \dots, X_{\ell})$ *is a polynomial over* $\mathbb{K}$ *that can be efficiently obtained from the commitment key* ck*;*
- $p_{\mathsf{sc}}(X_1, \dots, X_{\ell}) := f_{\mathsf{CM}}(P_{\mathsf{m}}(X_1, \dots, X_{\ell}), P_{\mathsf{ck}}(X_1, \dots, X_{\ell}), c)$ *is a polynomial over* $\mathbb{C}$*.*

Letting ideg *denote the maximum individual degree of a polynomial, we also define the following degrees:*

$$d_{\mathsf{ck}} := \max_{\mathsf{m} \in \mathbb{M}_{\mathsf{ck}}} \mathrm{ideg}\Big(P_{\mathsf{m}}(\underline{X})\Big) \ ,$$
$$d^{\star}_{\mathsf{ck}} := \max_{\mathsf{m} \in \mathbb{M}_{\mathsf{ck}}} \max_{c} \mathrm{ideg}\Big(f_{\mathsf{CM}}(P_{\mathsf{m}}(\underline{X}), P_{\mathsf{ck}}(\underline{X}), c)\Big) \ .$$

When running sumcheck arguments to prove knowledge of openings for certain sumcheck-friendly commitment schemes, such as Pedersen commitments, the verifier complexity is dominated by the verifier's final checks. In subsequent sections, we show how to delegate this computation and make the verifier succinct.

Definition 3.17. *The indexed relation* $\mathcal{R}_{\mathrm{SCA}}(\mathsf{pp}, c, B_{\mathsf{c}})$ *is the set of tuples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = \Big(\ \bot, \mathsf{cm}, \mathsf{m}\Big)$$

where CM *is a sumcheck-friendly commitment scheme,* $\mathsf{pp} = (\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck})$ *with* $\mathsf{pp}_{\mathsf{CM}} \in \mathsf{CM.Setup}(1^{\lambda}, n)$ *(that in particular specifies* $R, \mathcal{C}, \mathcal{H}, \ell, \mathbb{M}, \mathbb{K}, \mathbb{C}, f_{\mathsf{CM}}, \phi_{\mathrm{sc}}, \alpha_{\mathrm{sc}}$*),* $\mathsf{ck} \in \mathsf{CM.KeyGen}(\mathsf{pp}_{\mathsf{CM}}, B_{\mathsf{ck}})$*,* $\mathsf{cm} \in \mathbb{C}$*,* $\mathsf{m} \in \mathbb{M}_{\mathsf{ck}}$*,* $\|P_{\mathsf{m}}(\underline{X})\|_{\mathbb{M}} \leq B_{\mathsf{c}}$*,* $c \in \mathbb{S}_{\mathsf{ck}}$*,* $\mathsf{CM.Open}(\mathsf{ck}, \mathsf{m}, \mathsf{cm}, c) = 1$*.*

Construction 3.18 (sumcheck argument)**.** We describe a public-coin interactive argument $\mathsf{SCA} = (\mathbf{P}, \mathbf{V})$ for the relation in Definition 3.17. The prover $\mathbf{P}$ and verifier $\mathbf{V}$ take as input public parameters $\mathsf{pp} = (\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck})$, and an instance $\mathbb{x} = \mathsf{cm}$; the prover $\mathbf{P}$ additionally takes as input a witness $\mathbb{w} = \mathsf{m}$. Here, CM is sumcheck-friendly with respect to the ring R, subset $\mathcal{H}$ and challenge set $\mathcal{C}$.

The prover $\mathbf{P}$ and verifier $\mathbf{V}$ engage in a sumcheck protocol (described in Protocol 1) for the instance

$$\mathbb{x}_{\mathrm{SC}} := \Big(R, M = \mathbb{C}, \mathcal{H}, \ell = \log n, \tau = \mathsf{cm}, \mathcal{C}\Big)$$

where the prover $\mathbf{P}$ uses the polynomial $p_{\mathsf{sc}}(\underline{X}) := f_{\mathsf{CM}}(P_{\mathsf{m}}(\underline{X}), P_{\mathsf{ck}}(\underline{X}), 1)$ induced by $\mathbb{x}$ and $\mathbb{w}$.

After the end of the sumcheck protocol, the prover $\mathbf{P}$ learns $\underline{r} \in \mathcal{C}^{\ell}$ and the verifier $\mathbf{V}$ learns $(\underline{r}, v) \in \mathcal{C}^{\ell} \times \mathbb{C}$. (If the sumcheck verifier rejects, then $\mathbf{V}$ rejects.) Then the prover $\mathbf{P}$ computes and sends $w := P_{\mathsf{m}}(\underline{r}) \in \mathbb{M}$ to the verifier $\mathbf{V}$. **Final verifier checks:** The verifier $\mathbf{V}$ checks that $\|w\|_{\mathbb{M}} \leq B_{\mathsf{C}} \cdot (d_{\mathsf{ck}}+1)^{\ell}\, \gamma_R^{\ell d_{\mathsf{ck}}}\, \|\mathcal{C}\|_R^{\ell d_{\mathsf{ck}}}$, computes $P_{\mathsf{ck}}(\underline{r}) \in \mathbb{K}$, checks that $f_{\mathsf{CM}}(w, P_{\mathsf{ck}}(\underline{r}), 1) = v$, and checks that $\alpha_{\mathsf{sc}}(\mathsf{ck}, w, \underline{r}) = 1$.

Theorem 3.19. *The sumcheck argument* SCA *in Construction 3.18 satisfies the following properties:*

- *If* CM *is sumcheck-friendly (see Definition 3.16), then Construction 3.18 has perfect completeness.*
- *If* CM *is invertible with parameter K (see Definition A.1), then Construction 3.18 has K^{ℓ}-tree extraction.*
- *Communication: the prover sends $(d^{\star}_{\mathsf{ck}}+1)\ell$ elements of $\mathbb{C}$, an element of $\mathbb{M}$ with norm at most $B_{\mathsf{C}} \cdot (d_{\mathsf{ck}}+1)^{\ell}\, \gamma_R^{\ell d_{\mathsf{ck}}}\, \|\mathcal{C}\|_R^{\ell d_{\mathsf{ck}}}$, and an element of $\mathbb{R}$, and the verifier sends ℓ elements of $\mathcal{C}$.*
- *The prover performs the following operations: computing $p_{\mathsf{sc}}(\underline{X})$, and partially evaluating it $O(|\mathcal{H}|^{\ell-1})$ times; and $O(d^{\star}_{\mathsf{ck}}|\mathcal{H}|^{\ell-1})$ additions and scalar-multiplications in $\mathbb{C}$.*
- *The verifier performs the following operations: $O(d^{\star}_{\mathsf{ck}}|\mathcal{H}|\ell)$ additions and scalar multiplications in $\mathbb{C}$; 1 evaluation of the polynomial $P_{\mathsf{ck}}(\underline{X})$; 1 evaluation of f_{CM}; one evaluation of α_{sc}; and a norm-check on w.*

3.5.1 Linear-function commitments and Polynomial commitments

We restate the constructions of linear-function commitments and polynomial commitments using sumcheck arguments of [BCS21], to account for the removal of definitions and parameters related to hiding.

Definition 3.20. *Let* $\mathsf{BM} = (\mathsf{Setup}, \mathsf{KeyGen})$ *be a secure and hiding-friendly bilinear-module generator. The* **linear-function commitment scheme** LF *is defined via the following algorithms.*

- $\mathsf{LF.Setup}(1^{\lambda}, n)$: *sample* $(\mathcal{M}, \mathsf{aux}) \leftarrow \mathsf{BM.Setup}(1^{\lambda}, n)$ *and output* $\mathsf{pp}_{\mathsf{CM}} := (\mathcal{M}, \mathsf{aux})$.
- $\mathsf{LF.KeyGen}(\mathsf{pp}_{\mathsf{CM}})$: *sample* $\mathsf{ck} \leftarrow \mathsf{BM.KeyGen}(\mathcal{M}, \mathsf{aux})$, *and output* $\mathsf{ck} \in M_{\mathrm{R}}^{n}$.
- $\mathsf{LF.Commit}(\mathsf{ck}, \mathsf{m})$: *given* $\mathsf{ck} \in M_{\mathrm{R}}^{n}$, $\mathsf{m} = (\mathsf{mP}, \mathsf{mS}) \in R^{n} \times M_{\mathrm{L}}^{n}(B_{\mathsf{C}})$ *output*

$$\mathsf{cm} := \Big(\mathsf{mP}, \langle \mathsf{mS}, \mathsf{ck}\rangle, \langle \mathsf{mP}, \mathsf{mS}\rangle\Big) \in R^{n} \times M_{\mathrm{T}} \times M_{\mathrm{L}} \ .$$

- $\mathsf{LF.Open}(\mathsf{ck}, \mathsf{m}, \mathsf{cm}, o, c)$: *check that* $\mathsf{ck} \in M_{\mathrm{R}}^{n}$, $\mathsf{m} = (\mathsf{mP}, \mathsf{mS}) \in R^{n} \times M_{\mathrm{L}}^{n}(B_{\mathsf{BRA}})$, $o := \bot$, *and* $c \in R$, *such that* $c \cdot \mathsf{cm} = (c \cdot \mathsf{mP}, \langle \mathsf{mS}, \mathsf{ck}\rangle, \langle \mathsf{mP}, \mathsf{mS}\rangle)$.

The properties of sumcheck arguments for linear-function commitments follow from [BCS21, Theorem 4.6, Lemma 5.13]. Finally, [BCS21, Theorem B.4] gives the running time of the prover and the verifier.

Theorem 3.21. *Assuming that* BM *is secure,* LF *is a computationally binding commitment scheme such that*

- *the commitment key size is n elements of M_{R};*
- *computing* $\mathsf{LF.Commit}$ *requires n applications of e and $O(n)$ operations in $M_{\mathrm{L}}(B_{\mathsf{C}} + \|\mathsf{mP}\|_R)$ and M_{T}.*

Let $B^{\star} := B_{\mathsf{C}} \cdot n \gamma_R^{\log n}\, \|\mathcal{C}\|_R^{\log n}$, $D_{\mathsf{LF}} := 6\gamma_R^2 D^3 \iota(\mathcal{C}, 3)\|\mathcal{C}\|_R$, and $\xi_{\mathsf{LF}} := \xi^{3\log n}$. The sumcheck argument SCA *for linear-function commitments satisfies the following properties:*

- ***Perfect completeness***
- ***Knowledge soundness:*** *If $B_{\mathsf{BRA}} \geq B^\star \cdot D_{\mathsf{LF}}^{\log n}$, then the sumcheck argument has witness-extended emulation for $\mathcal{R}_{\mathrm{SCA}}(\xi_{\mathsf{LF}}, B^\star \cdot D_{\mathsf{LF}}^{\log n})$ with soundness error $\epsilon = \frac{\log n}{|\mathcal{C}|} + \mathrm{negl}(\lambda)$.*
- ***Communication:*** *the prover sends* mP*,* $O(\log n)$ *elements of* $M_{\mathrm{T}} \times M_{\mathrm{L}}(B_{\mathsf{c}})$ *and an element of* $M_{\mathrm{L}}(B^\star)$*, and the verifier sends* $\log n$ *elements of* $\mathcal{C}$*.*
- ***Prover Efficiency:*** *the prover can be implemented in* $O(n)$ *operations in* M_{R}*,* M_{T}*, and* $M_{\mathrm{L}}(B^\star)$*; and* $O(n)$ *applications of* e*.*
- ***Verifier Efficiency:*** *the verifier performs* $O(\log n)$ *operations in* M_{T} *and* $M_{\mathrm{L}}(B^\star)$*;* 1 *application of* e*; one norm-check over* M_{L}*; one evaluation of a multilinear polynomial over* M_{R} *with* $\log n$ *variables.*

Note that in [BCS21] the verifier running time is $O(n)$ due to the final verifier checks that require the evaluation of a multilinear polynomial. We will be interested in cases where mP has a short description, in which case the prover only sends the short description $\ll \mathsf{mP} \gg$ instead of mP.

Now, we are ready to define a polynomial commitment scheme for multivariate polynomials represented using the vector $\underline{P}$ of coefficients.

Construction 3.22. Let $\mathsf{BM} = (\mathsf{Setup}, \mathsf{KeyGen})$ be a bilinear-module generator. The polynomial commitment PC is defined via the following algorithms.

- $\mathsf{PC.Setup}(1^\lambda, n)$*:* sample $(\mathcal{M}, \mathsf{aux}) \leftarrow \mathsf{BM.Setup}(1^\lambda, n)$ and output $\mathsf{pp}_{\mathsf{CM}} := (\mathcal{M}, \mathsf{aux})$.
- $\mathsf{PC.KeyGen}(\mathsf{pp}_{\mathsf{CM}}, n)$*:* sample $\mathsf{ck} \leftarrow \mathsf{BM.KeyGen}(\mathcal{M}, \mathsf{aux})$, and output $\mathsf{ck} \in M_{\mathrm{R}}^n$.
- $\mathsf{PC.Commit}(\mathsf{ck}, \underline{P})$*:* given $\underline{P} \in M_{\mathrm{L}}^n(B_{\mathsf{c}})$, output $\mathsf{cm} := \langle \underline{P}, \mathsf{ck} \rangle$
- $\mathsf{PC.Open}(\mathsf{ck}, \underline{P}, \mathsf{cm}, o, c)$*:* check that $\mathsf{ck} \in M_{\mathrm{R}}^n$, $\underline{P} \in M_{\mathrm{L}}^n(B_{\mathsf{BRA}})$, and $c \in R$ such that $c \cdot \mathsf{cm} = \langle \underline{P}, \mathsf{ck} \rangle$.
- The interactive argument for $\mathcal{R}_{\mathrm{PC}}$, $\mathsf{PC\text{-}Eval} = (\mathbf{G}, \mathbf{I}, \mathbf{P}, \mathbf{V})$ is as follows:
  - $\mathbf{G}$ outputs $\mathsf{pp}_{\mathsf{CM}} \leftarrow \mathsf{PC.Setup}(1^\lambda)$;
  - $\mathbf{I}$ outputs $\perp$;
  - $\mathbf{P}$ and $\mathbf{V}$ run a sumcheck argument for the linear-function commitment $(\bigotimes_{i=1}^{\log n}(1, z_i), \mathsf{cm}, v)$, where the short description of $\ll \bigotimes_{i=1}^{\log n}(1, z_i) \gg = (z_1, \ldots, z_{\log n})$, to show that $\mathsf{cm} = \langle \underline{P}, \mathsf{ck} \rangle$ and $v = \langle \bigotimes_{i=1}^{\log n}(1, z_i), \underline{P} \rangle := P(z_1, \ldots, z_{\log n})$.

As with the linear function commitment scheme, the $\mathsf{PC\text{-}Eval}$ argument from Construction 3.22 suffers $O(n)$ verification time. However, it will later be used as part of the opening algorithm for the polynomial commitment scheme with $O(\log^2 n)$ verification costs.

4 Leveled bilinear module systems

We define leveled bilinear modules, a generalization of bilinear modules (Definition 3.11).

Definition 4.1. *A K-**level bilinear-module system** $\underline{\mathcal{M}}$ is a tuple*

$$\Big((\mathcal{M}_i, B_i)_{i=1}^{K}, (\mathsf{up}_i, \mathsf{dn}_i, \delta_{i+1})_{i=1}^{K-1}, \Big)$$

where:

- *each $\mathcal{M}_i := (R, M_{\mathrm{L},i}, M_{\mathrm{R},i}, M_{\mathrm{T},i}, e_i)$ is a bilinear module;*
- *each $B_i \in \mathbb{N}$ is a norm bound;*
- *each $\mathsf{up}_i \colon M_{\mathrm{R},i} \to M_{\mathrm{L},i+1}^{\delta_{i+1}}(B_{i+1})$ is an "upward map";*
- *each $\mathsf{dn}_i \colon M_{\mathrm{L},i+1}^{\delta_{i+1}} \to M_{\mathrm{R},i}$ is a "downward map";*
- *each $\delta_{i+1} \in \mathbb{N}$ is an "expansion constant".*

The tuple satisfies the following for every $i \in [K-1]$

- *$\mathsf{dn}_i \circ \mathsf{up}_i$ is the identity map on $M_{\mathrm{R},i}$;*
- *$\mathsf{up}_i \colon M_{\mathrm{R},i} \to M_{\mathrm{L},i+1}^{\delta_{i+1}}$ is an R-module isomorphism i.e. for every $m_1, m_2 \in M_{\mathrm{R},i}$, $\mathsf{up}_i(m_1 + m_2) = \mathsf{up}_i(m_1) + \mathsf{up}_i(m_2) \bmod \ker \mathsf{dn}_i$, and for every $r \in R$, $m \in M_{\mathrm{R},i}$, $\mathsf{up}_i(r \cdot m) = r \cdot \mathsf{up}_i(m) \bmod \ker \mathsf{dn}_i$.*

Remark 4.2. For a vector $\underline{m} = (m_1, \ldots, m_n) \in M_{\mathrm{R},i}^n$, we denote by $\mathsf{up}_i(\underline{m}) := (\underline{x}_1, \ldots, \underline{x}_{\delta_{i+1}}) \in (M_{\mathrm{L},i+1}^n(B_{i+1}))^{\delta_{i+1}}$ the vector such that $\mathsf{up}_i(m_j) = (x_{1,j}, \ldots, x_{\delta_{i+1},j})$ for each $j \in [n]$, where $\underline{x}_k := (x_{k,1}, \ldots, x_{k,n})$ for $k \in [\delta_{i+1}]$.

The following lemma shows how to compress δ_i elements of $M_{\mathrm{L},i}$ to a single "random" $M_{\mathrm{L},i}$ element.

Lemma 4.3. *For every $\underline{m} \in M_{\mathrm{L},i+1}^{\delta_{i+1}} \setminus \{0^{\delta_{i+1}}\}$, $\Pr_{\underline{\rho} \leftarrow \{0,1\}^{\delta_{i+1}}}[\langle \underline{\rho}, \underline{m} \rangle = 0_{M_{\mathrm{L},i+1}}] \leq 1/2$.*

Proof. Since $\underline{m} := (m_1, \ldots, m_{\log q}) \in M_{\mathrm{L},i+1}^{\delta_{i+1}} \setminus \{0^{\delta_{i+1}}\}$, there is a non-zero module element in $\underline{m}$. Let $m_j \neq 0_{M_{\mathrm{L},i+1}}$, then

$$\begin{aligned}
&\Pr_{\underline{\rho} \leftarrow \{0,1\}^{\delta_{i+1}}}\left[\langle \underline{\rho}, \underline{m} \rangle = 0_{M_{\mathrm{L},i+1}}\right] = \Pr_{\underline{\rho} \leftarrow \{0,1\}^{\delta_{i+1}}}\left[\rho_j \cdot m_j = -\sum_{k \neq j} \rho_k m_k\right] \\
&= \Pr_{\rho_j \leftarrow \{0,1\}}\left[\rho_j \cdot m_j = 0 \,\middle|\, \sum_{k \neq j} \rho_k m_k = 0\right] \cdot \Pr_{\underline{\rho} \leftarrow \{0,1\}^{\delta_{i+1}}}\left[\sum_{k \neq j} \rho_k m_k = 0\right] \\
&+ \Pr_{\rho_j \leftarrow \{0,1\}}\left[\rho_j \cdot m_j = m_j \,\middle|\, \sum_{k \neq j} \rho_k m_k = -m_j\right] \cdot \Pr_{\underline{\rho} \leftarrow \{0,1\}^{\delta_{i+1}}}\left[\sum_{k \neq j} \rho_k m_k = -m_j\right] \\
&\leq 1/2 \left(\Pr_{\rho \leftarrow \{0,1\}^{\delta_{i+1}}}\left[\sum_{k \neq j} \rho_k m_k = 0\right] + \Pr_{\underline{\rho} \leftarrow \{0,1\}^{\delta_{i+1}}}\left[\sum_{k \neq j} \rho_k m_k = -m_j\right]\right) \\
&\leq 1/2.
\end{aligned}$$

□

Definition 4.4. *A* **leveled bilinear module generator** *is a tuple* $\mathsf{LM} = (\mathsf{LM.Setup}, \mathsf{LM.KeyGen})$ *with the following syntax:*

- $\mathsf{LM.Setup}$ *receives as input a security parameter* $\lambda \in \mathbb{N}$ *(represented in unary), a number of levels* $K \in \mathbb{N}$*, and a size bound* $N \in \mathbb{N}$*, and outputs a* K*-level bilinear-module system* $\underline{\mathcal{M}}$ *and an auxiliary string* aux*;*
- $\mathsf{LM.KeyGen}$ *receives as input a* K*-level bilinear-module system* $\underline{\mathcal{M}}$*, a vector* $\underline{n} \in [N]^K$*, and* aux*, and outputs vectors* $\underline{x}_1, \ldots, \underline{x}_K$ *with* $\underline{x}_i \in M_{\mathrm{R},i}^{n_i}$*.*

We assume without loss of generality that the parameters λ, K, N *are part of* aux*.*

The leveled bilinear module generators that we consider output auxiliary strings that contain several pieces of information:

$$\mathsf{aux} = ((B_{\mathsf{BRA},i}, \xi_i, D_i)_{i=1}^K, \mathcal{C})$$

where $B_{\mathsf{BRA},i} \in \mathbb{N}$ with $\gamma_R \cdot \delta_{i+1} \cdot B_i \leq B_{\mathsf{BRA},i}$, $\xi_i \in R$, $D_i \in \mathbb{Z}$, and $\mathcal{C} \subseteq R$.

The leveled bilinear relation assumption is the main cryptographic assumption that we use. We also define other useful properties for leveled bilinear module generators.

Definition 4.5. LM *satisfies the* **K-level bilinear relation assumption (K-LBRA)** *if for all* $N \in \mathbb{N}$*,* $\underline{n} \in [N]^K$*, predicate* Check*, and polynomial-size adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l|c} \mathsf{Check}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux}) = 1 & (\underline{\mathcal{M}}, \mathsf{aux}) \leftarrow \mathsf{LM.Setup}(1^\lambda, K, N) \\ \wedge\ \underline{y} \in M_{\mathrm{L},k}^{n_k}(B_{\mathsf{BRA},k}) & (\underline{x}_i)_{i=1}^K \leftarrow \mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux}) \\ \wedge\ \underline{y} \neq \{0^{n_k}\} & (k, \underline{y}) \leftarrow \mathcal{A}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux}, (\underline{x}_i)_{i=1}^K) \\ \wedge\ \langle \underline{y}, \underline{x_k} \rangle = 0 & \end{array}\right] \leq \epsilon(\lambda, N) \enspace .$$

The case $K = 1$ *corresponds to the bilinear relation assumption of [BCS21] (see Definition 3.14).*

Similarly to the case of bilinear modules, the "Check" algorithm in Definition 4.5 is used to determine whether the parameters of $\underline{\mathcal{M}}$ and aux are suitable for use as part of larger algorithms which may introduce additional constraints.

Definition 4.6. LM *is* **K-secure** *if: (a)* LM *satisfies the* **K-LBRA***; and (b) for every* $i \in [K]$*,* $(\mathcal{C}, \xi_i, D_i)$ *in* aux *are pseudoinverse parameters for* $(R, M_{\mathrm{T},i})$ *output by* $\mathsf{LM.Setup}$ *(Definition 3.12).*

Definition 4.7. LM *is* **elimination-friendly** *if, for every* $i \in [K]$*,* $(\mathcal{C}, \xi_i, D_i)$ *in* aux *are pseudoinverse parameters for* $(R, M_{\mathrm{T},i})$ *output by* $\mathsf{LM.Setup}$ *(Definition 3.12) and* ξ_i *is not a zero-divisor in* $M_{\mathrm{L},i}$*.*

Definition 4.8. LM *is* **quotient-friendly** *if, for every* $i \in [K]$*,* $M_{\mathrm{L},i}$ *output by* $\mathsf{LM.Setup}$ *contains an* R*-submodule* $I_i \subseteq M_{\mathrm{L},i}$ *such that* $(\mathcal{C}, \xi_i, D_i)$ *in* aux *output by* $\mathsf{LM.Setup}$ *are pseudoinverse parameters for* $(R, M_{\mathrm{T},i})$*, multiplication by* ξ_i *is invertible in* $M_{\mathrm{L},i}/I_i$ *and each element of* $M_{\mathrm{L},i}/I_i$ *has at least one representative in* $M_{\mathrm{L},i}(B_i)$*.*

4.1 Instantiations

We give five instantiations of leveled bilinear modules: one based on bilinear groups, and four based on ideal lattices. The challenge space used in the 2-power-cyclotomic instantiations originates from [BCKLN14] and is commonplace in lattice-based probabilistic proofs, whereas the challenge space used in the odd-prime-power instantiations comes from [AL21]. Of the following instantiations, the instantiation based on bilinear groups and the lattice-based instantiations incorporating bit decomposition can work for $K = \log n$ (what we need in this paper).

The two lattice-based instantiations without bit decomposition work for a constant K. These instantiations do *not* extend to any number of levels, because up_i must map elements of $M_{\mathrm{R},i} := R/q_iR$ to elements of $M_{\mathrm{L},i+1} := R$ for which the ring SIS assumption holds modulo q_i. This means that q_{i+1} must be sufficiently large compared to q_i, leading to successively larger moduli. However, the SIS problem is easy to solve when the modulus is too large.

	bilinear groups	cyclotomic rings ($d = 2^t$)	cyclotomic rings ($d = p^t$, $p > 2$ prime)	cyclotomic rings ($d = 2^t$)	cyclotomic rings ($d = p^t$, $p > 2$ prime)
$K = \omega(1)$?	✓	✗	✗	✓	✓
R	$\mathbb{F}_q$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$
$M_{\mathrm{L},i}$	$\mathbb{G}_{i \bmod 2}$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}[X]/\langle\Phi_d(X)\rangle$
$M_{\mathrm{R},i}$	$\mathbb{G}_{i+1 \bmod 2}$	$\mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle$
$M_{\mathrm{T},i}$	$\mathbb{G}_{\mathrm{T}}$	$\mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle$	$\mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle$
e_i	bilinear map	poly multiplication mod q_i and $\Phi_d(X)$	poly multiplication mod q_i and $\Phi_d(X)$	poly multiplication mod q and $\Phi_d(X)$	poly multiplication mod q and $\Phi_d(X)$
B_i	q_{i-1}	q_{i-1}	1	1	1
up_i	identity	inclusion	inclusion	bit decomposition	bit decomposition
dn_i	identity	mod q_i	mod q_i	bit composition	bit composition
δ_{i+1}	1	1	1	$\log q$	$\log q$
R-norm	trivial †	ℓ_∞	ℓ_∞	ℓ_∞	ℓ_∞
$M_{\mathrm{L},i}$-norm	trivial †	ℓ_∞	ℓ_∞	ℓ_∞	ℓ_∞
$B_{\mathsf{BRA},i}$	∞	$B_{\mathsf{SIS},i}$	$B_{\mathsf{SIS},i}$	B_{SIS}	B_{SIS}
$\mathcal{C}$	$\mathbb{F}_q$	$\{X^j : 0 \le j \le d-1\}$	$\left\{\frac{X^j-1}{X-1} : 0 \le j \le p-1\right\}$	$\{X^j : 0 \le j \le d-1\}$	$\left\{\frac{X^j-1}{X-1} : 0 \le j \le p-1\right\}$
ξ_i	1	2	1	2	1
D_i	1	$d/2$	$\phi(d)$	$d/2$	$\phi(d)$
I_i	$\{0\}$	$n\mathbb{Z}$ for odd $n \neq -1, 1$ ‡	$n\mathbb{Z}$ for any $n \neq -1, 1$	$n\mathbb{Z}$ for odd $n \neq -1, 1$ ‡	$n\mathbb{Z}$ for any $n \neq -1, 1$

Figure 1: Output $(\underline{\mathcal{M}}, \mathsf{aux})$ of a leveled bilinear-module generator in the different cryptographic settings, where $\mathcal{M}_i = (R, M_{\mathrm{L},i}, M_{\mathrm{R},i}, M_{\mathrm{T},i}, e_i)$ and $\mathsf{aux} = ((I_i, B_{\mathsf{BRA},i}, \xi_i, D_i)_{i=1}^{K}, \mathcal{C})$. (†: Equals 1 for any non-zero element of R or $M_{\mathrm{L},i}$ and equals 0 otherwise. ‡: As in [BCS21])

Bilinear groups. The algorithm $\mathsf{LM.Setup}(1^\lambda, K, N)$ samples groups $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_{\mathrm{T}}$ of prime order $q \approx 2^\lambda$ equipped with a bilinear (pairing) map $e\colon \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_{\mathrm{T}}$ and outputs $(\underline{\mathcal{M}}, \mathsf{aux})$ as in Figure 1. The algorithm $\mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux})$ samples uniformly random vectors $\underline{\mathsf{G}}_1, \ldots, \underline{\mathsf{G}}_K \in M_{\mathrm{R},i}^{n_i} = \mathbb{G}_{i+1 \bmod 2}^{n_i}$. We explain why LM is a K-secure, quotient-friendly, leveled bilinear module generator.

- LM is a leveled bilinear module generator, because for each i,
  - $\mathsf{dn}_i \circ \mathsf{up}_i$ is the identity on $\mathbb{G}_{i+1 \bmod 2}$;
  - for every $m_1, m_2 \in \mathbb{G}_{i+1 \bmod 2}$, $\mathsf{up}_i(m_1 + m_2) = \mathsf{up}_i(m_1) + \mathsf{up}_i(m_2)$;
  - for every $r \in \mathbb{F}_q$, $m \in \mathbb{G}_{i+1 \bmod 2}$, $\mathsf{up}_i(r \cdot m) = r \cdot \mathsf{up}_i(m)$.

  This implies that $\underline{\mathcal{M}}$ is a K-level bilinear-module system.

- LM is K-secure if the K-level bilinear relation assumption holds for $\mathsf{LM} = (\mathsf{LM.Setup}, \mathsf{LM.KeyGen})$. This translates to the assumptions that given a uniformly random vector $\underline{\mathsf{H}} \in \mathbb{G}_1^n$ (resp. $\underline{\mathsf{G}} \in \mathbb{G}_0^n$), finding $\underline{a} \in \mathbb{G}_0^n$ such that $\langle \underline{a}, \underline{\mathsf{H}} \rangle = 0$ (resp. $\underline{b} \in \mathbb{G}_1^n$ such that $\langle \underline{\mathsf{G}}, \underline{b} \rangle = 0$) is a computationally intractable problem. These assumptions reduce to the *double pairing assumption* and *reverse double pairing assumption* respectively, and both are implied by the SXDH assumption [AFGHO16].

- To see that LM is quotient friendly, observe that $(\mathcal{C}, \xi_i, D_i) := (\mathbb{F}_q, 1, 1)$ are pseudoinverse parameters for $(R, M_{\mathrm{T},i}) := (\mathbb{F}_q, \mathbb{G}_{\mathrm{T}})$ since for every $a \in \mathbb{F}_q$, $m, m^* \in \mathbb{G}_{\mathrm{T}}$, and distinct $c_1, c_2 \in \mathbb{F}_q$ it holds that if

$(c_1 - c_2)m = a \cdot m^*$, then $m = r \cdot m^*$ for $r = (c_1 - c_2)^{-1}a \in \mathbb{F}_q$. Further, multiplication by $\xi_i = 1$ is invertible in $M_{\mathrm{L},i}/I_i = \mathbb{G}_{i \bmod 2}$.

Cyclotomic rings with $d = 2^t$ and constant K. The algorithm $\mathsf{LM.Setup}(1^\lambda, K, N)$ samples an integer d that is a power of 2, and odd positive integers $(q_i)_{i=1}^K$ such that $q_j \leq q_{j+1}$ for $j = 1, \ldots, K-1$, and outputs $(\underline{\mathcal{M}}, \mathsf{aux})$ as in Figure 1; note that the challenge set $\mathcal{C}$ consists of d elements. The algorithm $\mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux})$ outputs uniformly random vectors in $M_{\mathrm{R},i}^{n_i} = (\mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle)^{n_i}$. We explain why LM is a K-secure, quotient-friendly, leveled bilinear module generator.

- LM is a leveled bilinear module generator, because for each i,
  - $\mathsf{dn}_i \circ \mathsf{up}_i$ is the identity on $\mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle$,
  - $\forall m_1, m_2 \in \mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle$; $\mathsf{up}_i(m_1 + m_2) = \mathsf{up}_i(m_1) + \mathsf{up}_i(m_2) \bmod q_i$;
  - $\forall r \in \mathbb{Z}[X]/\langle\Phi_d(X)\rangle$, $m \in \mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle$, $\mathsf{up}_i(r \cdot m) = r \cdot \mathsf{up}_i(m) \bmod q_i$.

  This implies that $\underline{\mathcal{M}}$ is a K-level bilinear-module system.

- LM is K-secure if the K-level bilinear relation assumption holds for $\mathsf{LM} = (\mathsf{LM.Setup}, \mathsf{LM.KeyGen})$. This translates to the hardness of the SIS assumption for $\mathcal{M}_i$ with norm bounds $B_{\mathsf{SIS},i}$ for $i = 1, \ldots, K$. In turn, for the SIS assumption to hold, $B_{\mathsf{SIS},i}$ should be at most $\min\{q_i, 2^{2\sqrt{d \log q_i \log \delta}}\}$ [GN08] (δ is related to the optimal block size in the BKZ algorithm applied to the SIS problem and is typically set to $\delta \approx 1.005$).

- To see that LM is quotient friendly, note that $\Phi_d(X) = X^{d/2} + 1$. Now, $(\mathcal{C}, \xi_i, D_i) := (\mathcal{C}, 2, d/2)$ are pseudoinverse parameters for $(R, M_{\mathrm{T},i}) := (\mathbb{Z}[X]/\langle X^d + 1\rangle, \mathbb{Z}_q[X]/\langle X^{d/2} + 1\rangle)$ since for every $a \in \mathbb{Z}[X]/\langle X^{d/2} + 1\rangle$, $m, m^* \in \mathbb{Z}_q[X]/\langle X^{d/2} + 1\rangle$, and distinct $X^{c_1}, X^{c_2} \in \mathcal{C}$ it holds that if $(X^{c_1} - X^{c_2})m = a \cdot m^*$, then $2 \cdot m = r \cdot m^*$ for $r = -X^{d/2 - c_1}(1 + X + \cdots + X^{d/2}) \cdot a \in \mathbb{Z}[X]/\langle X^{d/2} + 1\rangle$ [BCKLN14]. Further, multiplication by $\xi_i = 2$ is invertible in $M_{\mathrm{L},i}/I_i$.

Cyclotomic rings with $d = p^t$ and constant K. The algorithm $\mathsf{LM.Setup}(1^\lambda, K, N)$ samples an integer d that is an odd prime power, and numbers $(q_i)_{i=1}^K$ such that $q_j \leq q_{j+1}$ for $j = 1, \ldots, K-1$, and outputs $(\underline{\mathcal{M}}, \mathsf{aux})$ as in Figure 1; note that the challenge set $\mathcal{C}$ consists of p elements. The algorithm $\mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux})$ outputs uniformly random vectors in $M_{\mathrm{R},i}^{n_i} = (\mathbb{Z}_{q_i}[X]/\langle\Phi_d(X)\rangle)^{n_i}$. The leveled-module generator is K-secure, using similar reasoning to the previous example.

To see that LM is quotient friendly, note that $(\mathcal{C}, \xi_i, D_i) := (\mathcal{C}, 1, \phi(d))$ are pseudoinverse parameters for $(R, M_{\mathrm{T},i}) := (\mathbb{Z}[X]/\langle\Phi_d(X)\rangle, \mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle)$ ([AL21, Theorem 2]), and that multiplication by $\xi_i = 1$ is invertible in $M_{\mathrm{L},i}/I_i$.

Cyclotomic rings with $d = 2^t$ and non-constant K. The algorithm $\mathsf{LM.Setup}(1^\lambda, K, N)$ samples an integer d that is a power of 2, and an odd positive integer q, and outputs $(\underline{\mathcal{M}}, \mathsf{aux})$ as in Figure 1 with the same challenge space and $\mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux})$ as the previous $d = 2^t$ example. We define bit decomposition $\mathrm{BitDec} : \mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle \to R^{\log q}$ to be the function that maps a ring element $m \in \mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle$ to $(m'_1, \ldots, m'_{\log q})$ such that each coefficient of m'_i is equal to the i-bit of the corresponding coefficient of m. We define bit composition $\mathrm{BitComp} : R^{\log q} \to \mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle$ to be the function that maps ring elements $(m'_1, \ldots, m'_{\log q}) \in (\mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle)^{\log q}$ to $(\sum_{i=0}^{\log q - 1} 2^i m'_i) \bmod q$. We explain why LM is a K-secure, quotient-friendly, leveled bilinear module generator.

- LM is a leveled bilinear module generator, because for each i,
  - $\mathsf{dn}_i \circ \mathsf{up}_i$ is the identity on $\mathbb{Z}_q[X]/\langle\Phi_d(X)\rangle$;

- since $\mathsf{dn}_i := \mathrm{BitComp}$ is a linear map, $\forall m_1, m_2 \in \mathbb{Z}_q[X]/\langle \Phi_d(X) \rangle$, $\mathsf{dn}_i(\mathsf{up}_i(m_1) + \mathsf{up}_i(m_2)) = \mathsf{dn}_i(\mathsf{up}_i(m_1)) + \mathsf{dn}_i(\mathsf{up}_i(m_2)) = m_1 + m_2$. Hence,

  $$\mathsf{up}_i(m_1 + m_2) = \mathsf{up}_i(m_1) + \mathsf{up}_i(m_2) \bmod \ker \mathsf{dn}_i,$$

  where $\ker \mathsf{dn}_i = \{\underline{v} \in (\mathbb{Z}_q[X]/\langle \Phi_d(X) \rangle)^{\log q} : \mathrm{BitComp}(\underline{v}) = 0_R\}$;
- similarly, we can show that $\forall r \in \mathbb{Z}[X]/\langle \Phi_d(X) \rangle$, $m \in \mathbb{Z}_q[X]/\langle \Phi_d(X) \rangle$, $\mathsf{up}_i(r \cdot m) = r \cdot \mathsf{up}_i(m) \bmod \ker \mathsf{dn}_i$;

This implies that $\underline{\mathcal{M}}$ is a K-level bilinear-module system. The leveled-module generator is K-secure and quotient-friendly, using similar reasoning to the previous example.

Cyclotomic rings with $d = p^t$ and non-constant K. The algorithm $\mathsf{LM.Setup}(1^\lambda, K, N)$ samples an integer d that is an odd prime power, and a number q, and outputs $(\underline{\mathcal{M}}, \mathsf{aux})$ as in Figure 1 with the same challenge space and $\mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux})$ as the previous $d = p^t$ example.

The leveled-module generator is K-secure and quotient-friendly, using similar reasoning to the previous examples.

5 Delegation of multilinear polynomial evaluation

We describe a (preprocessing) succinct interactive argument for the polynomial evaluation relation. Recall from Definition 3.5 that $p_{\underline{v}}$ is the multilinear polynomial with coefficients given by the vector $\underline{v} \in M^n$.

Definition 5.1. *The indexed relation for the* **polynomial evaluation relation** $\mathcal{R}_{\mathsf{EVAL}}(\mathsf{pp})$ *consists of tuples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = (\underline{\mathsf{h}}, (B_s, \underline{s}, u), \emptyset)$$

where $\mathsf{pp} = (\underline{\mathcal{M}}, \mathsf{aux}) \in \mathsf{LM.Setup}(1^\lambda, \log n, n)$*,* $\underline{\mathsf{h}} \in M_{\mathrm{L},1}(B_1)^n$*,* $B_s \in \mathbb{R}_{>0}$ $\underline{s} \in R^{\log n}(B_s)$*, and* $u \in M_{\mathrm{L},1}$ *satisfying* $p_{\underline{\mathsf{h}}}(\underline{s}) = u$*.*

Later, in Section 6, we will always use instances $\mathbb{x}$ for which s lies in $\mathcal{C}^n$, where $\mathcal{C}$ is the challenge space associated with a levelled bilinear module. Therefore, we will set $B_s = \|\mathcal{C}\|_R$ from now on.

Construction 5.2. First we describe the generator and indexer, and then the interaction between the prover and verifier.

Generator. $\mathbf{G}_{\mathsf{EVAL}}(1^\lambda, n)$ works as follows.

1. Sample $(\underline{\mathcal{M}}, \mathsf{aux}) \leftarrow \mathsf{LM.Setup}(1^\lambda, \ell, n)$ with $\ell = \log n$ and

$$\underline{\mathcal{M}} = \left((\mathcal{M}_i, B_i)_{i=1}^{\ell}, (\mathsf{up}_i, \mathsf{dn}_i, \delta_{i+1})_{i=1}^{\ell-1},\right).$$

2. Output $\mathsf{pp} = (\underline{\mathcal{M}}, \mathsf{aux})$, where aux contains a description of a challenge space $\mathcal{C}$.

Indexer. $\mathbf{I}_{\mathsf{EVAL}}(\mathsf{pp}, \mathbb{i})$ works as follows.

1. Check that $\mathbb{i} = \underline{\mathsf{h}}$ is compatible with pp (i.e. $\underline{\mathsf{h}} \in M^n_{\mathrm{L},1}(B_1)$).
2. Sample $(\underline{\mathsf{H}}_i)_{i=1}^{\ell} \leftarrow \mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux})$, where $n_i = n/2^i$.
3. Parse $\underline{\mathsf{h}} = (\underline{\mathsf{h}}[L], \underline{\mathsf{h}}[R]) \in M^{n/2}_{\mathrm{L},1} \times M^{n/2}_{\mathrm{L},1}$.
4. Compute $\mathsf{cmL}_1 := \langle \underline{\mathsf{h}}[L], \underline{\mathsf{H}}_1 \rangle \in M_{\mathrm{T},1}$ and $\mathsf{cmR}_1 := \langle \underline{\mathsf{h}}[R], \underline{\mathsf{H}}_1 \rangle \in M_{\mathrm{T},1}$.
5. For $i \in \{1, \dots, \ell\}$, parse $\underline{\mathsf{H}}_i = (\underline{\mathsf{H}}_i[L], \underline{\mathsf{H}}_i[R]) \in M^{n_i/2}_{\mathrm{R},i} \times M^{n_i/2}_{\mathrm{R},i}$ and compute[5]

$$\begin{aligned}
&(\underline{\mathsf{h}}_{i+1,1}[L], \dots, \underline{\mathsf{h}}_{i+1,\delta_{i+1}}[L]) := \mathsf{up}_i(\underline{\mathsf{H}}_i[L]) \in (M^{n_i}_{\mathrm{L},i+1}(B_{i+1}))^{\delta_{i+1}}\\
&(\underline{\mathsf{h}}_{i+1,1}[R], \dots, \underline{\mathsf{h}}_{i+1,\delta_{i+1}}[R]) := \mathsf{up}_i(\underline{\mathsf{H}}_i[R]) \in (M^{n_i}_{\mathrm{L},i+1}(B_{i+1}))^{\delta_{i+1}}\\
&\mathsf{cmL}_{i+1,j} := \langle \underline{\mathsf{h}}_{i+1,j}[L], \underline{\mathsf{H}}_{i+1} \rangle \in M_{\mathrm{T},i+1} \text{ for } j \in [\delta_{i+1}]\\
&\mathsf{cmR}_{i+1,j} := \langle \underline{\mathsf{h}}_{i+1,j}[R], \underline{\mathsf{H}}_{i+1} \rangle \in M_{\mathrm{T},i+1} \text{ for } j \in [\delta_{i+1}]
\end{aligned}$$

6. Output $(\mathsf{ipk}, \mathsf{ivk})$ where $\mathsf{ipk} := \left(\mathsf{pp}, (\underline{\mathsf{H}}_i, (\mathsf{cmL}_{i,j}, \mathsf{cmR}_{i,j})_{j=1}^{\delta_{i+1}})_{i=1}^{\ell+1}\right)$ and $\mathsf{ivk} := \left(\mathsf{pp}, (\mathsf{cmL}_{i,j}, \mathsf{cmR}_{i,j})_{j=1}^{\delta_{i+1}})_{i=1}^{\ell+1}, \underline{\mathsf{H}}_\ell\right)$.

The interactive phase. We define $\underline{r}_0 := \underline{s} \in R^\ell(B_s)$, $u_1 := u \in M_{\mathrm{L},1}$, and $\underline{\mathsf{h}}_1 := \underline{\mathsf{h}} \in M^n_{\mathrm{L},i}(B_1)$. The prover $\mathbf{P}_{\mathsf{EVAL}}$ takes as input $(\mathsf{ipk}, \mathbb{x}, \mathbb{w})$ and the verifier $\mathbf{V}_{\mathsf{EVAL}}$ takes as input $(\mathsf{ivk}, \mathbb{x})$. They interact as follows. For $i = 1, \dots, \ell$:

[5]For notation of $\mathsf{up}_i(\underline{\mathsf{H}}_i[L])$ and $\mathsf{up}_i(\underline{\mathsf{H}}_i[R])$, see Remark 4.2

1. The prover $\mathbf{P}_{\mathsf{EVAL}}$ and the verifier $\mathbf{V}_{\mathsf{EVAL}}$ engage in a proof of polynomial evaluation as in Construction 3.22 (without the final verifier checks) for the claim

$$P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{r}'_{i-1}) = u_i,$$

where if $\underline{r}_{i-1} := (r_{i-1,1}, \ldots, r_{i-1,\ell-i+1})$, then $\underline{r}'_{i-1} := (r_{i-1,2}, \ldots, r_{i-1,\ell-i+1})$, and the commitment of $\underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R]$ is computed as $\mathsf{cmL}_i + r_{i-1,1} \cdot \mathsf{cmR}_i$.

Namely, the prover $\mathbf{P}_{\mathsf{EVAL}}$ and the verifier $\mathbf{V}_{\mathsf{EVAL}}$ engage in a sumcheck protocol for the polynomial

$$p^{(i)}_{\text{SC}}(\underline{X}) := \left(P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{X})P_{\underline{\mathsf{H}}_i}(\underline{X})\ ,\ P_{\underline{\tilde{r}}_{i-1}}(\underline{X})P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{X})\right)\ , \tag{1}$$

where $\underline{\tilde{r}}_{i-1} := \bigotimes_{j=1}^{\ell-i}(1, r_{i-1,j+1})$. After the end of the sumcheck protocol, the prover $\mathbf{P}_{\mathsf{EVAL}}$ learns $\underline{r}_i \in \mathcal{C}^{\ell-i}$ and the verifier $\mathbf{V}_{\mathsf{EVAL}}$ learns $(\underline{r}_i, (v_{i,0}, v_{i,1})) \in \mathcal{C}^{\ell-i} \times (M_{\mathrm{T},i} \times M_{\mathrm{L},i})$. (If the sumcheck verifier rejects, then $\mathbf{V}_{\mathsf{EVAL}}$ rejects.) Then, the prover $\mathbf{P}_{\mathsf{EVAL}}$ computes and sends $w_i := P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{r}_i) \in M_{\mathrm{L},i}$ and $U_i := P_{\underline{\mathsf{H}}_i}(\underline{r}_i) \in M_{\mathrm{R},i}$[6].

2. The prover $\mathbf{P}_{\mathsf{EVAL}}$ also computes[7]

$$\begin{aligned}
&(\underline{\mathsf{h}}_{i+1,1}, \ldots, \underline{\mathsf{h}}_{i+1,\delta_{i+1}}) := \mathsf{up}_i(\underline{\mathsf{H}}_i) \in (M^{n_i}_{\mathrm{L},i+1}(B_{i+1}))^{\delta_{i+1}}\\
&u_{i+1,j} := P_{\underline{\mathsf{h}}_{i+1,j}}(\underline{r}_i) \in M_{\mathrm{L},i+1} \text{ for } j \in [\delta_{i+1}]\\
&\kappa_{i+1} := \mathsf{up}_i(U_i) - (u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}}) \in M^{\delta_{i+1}}_{\mathrm{L},i+1}\ .
\end{aligned}$$

The prover sends $(u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}}, \kappa_{i+1})$. (For $i = \ell$, the prover sends only U_i.)

3. The verifier $\mathbf{V}_{\mathsf{EVAL}}$, after receiving $(w_i, U_i, u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}}, \kappa_{i+1})$, sends $\underline{\rho}_{i+1} \leftarrow \{0,1\}^{\delta_{i+1}}$.

4. The prover $\mathbf{P}_{\mathsf{EVAL}}$ computes

$$\underline{\mathsf{h}}_{i+1} := \sum_{j=1}^{\delta_{i+1}} \rho_{i+1,j}\underline{\mathsf{h}}_{i+1,j} \in M^{n_i}_{\mathrm{L},i+1}\ , \tag{2}$$

$$\mathsf{cmL}_{i+1} := \langle \underline{\rho}_{i+1}, (\mathsf{cmL}_{i+1,1}, \ldots, \mathsf{cmL}_{i+1,\delta_{i+1}})\rangle \in M_{\mathrm{T},i+1}\ , \tag{3}$$

$$\mathsf{cmR}_{i+1} := \langle \underline{\rho}_{i+1}, (\mathsf{cmR}_{i+1,1}, \ldots, \mathsf{cmR}_{i+1,\delta_{i+1}})\rangle \in M_{\mathrm{T},i+1}\ , \tag{4}$$

$$\text{and } u_{i+1} := \langle \underline{\rho}_{i+1}, (u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}})\rangle (= P_{\underline{\mathsf{h}}_{i+1}}(\underline{r}_i)) \in M_{\mathrm{L},i+1}\ . \tag{5}$$

(For $i = \ell$, the prover does not perform this step.)

5. The verifier $\mathbf{V}_{\mathsf{EVAL}}$ computes $\tau_i := P_{\underline{\tilde{r}}_{i-1}}(\underline{r}_i) \in R$, and cmL_{i+1}, cmR_{i+1} and u_{i+1} as in Equations 3 to 5. The verifier checks whether

$$\begin{aligned}
&(v_{i,0}, v_{i,1}) = (w_i \cdot U_i, \tau_i \cdot w_i)\ ,\\
&\|w_i\|_{M_{\mathrm{L},i}} \le 2^{\ell-i}\gamma_R^{\ell-i+1}\|\mathcal{C}\|_R^{\ell-i}(1 + \gamma_R\|\mathcal{C}\|_R) \cdot \delta_i \cdot B_i\ ,\\
&\langle \underline{\rho}_{i+1}, (u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}})\rangle = \langle \underline{\rho}_{i+1}, \mathsf{up}_i(U_i) - \kappa_{i+1}\rangle\ ,\\
&\kappa_{i+1} \in \ker \mathsf{dn}_i\ .
\end{aligned}$$

[6] In sumcheck arguments, the prover computes and sends $p^{(i)}_{\text{SC}}(\underline{r}_i)$. In this case, the verifier already knows $\underline{\tilde{r}}_{i-1}$ and can efficiently compute $P_{\underline{\tilde{r}}_{i-1}}(\underline{r}_i)$ with $O(\log n)$ operations as shown in Section 5.1, so the prover only needs to send w_i.

[7] Note that because $\underline{\mathcal{M}}$ is a levelled bilinear-module system, $\kappa \in \ker \mathsf{dn}_i$.

6. The verifier $\mathbf{V}_{\mathsf{EVAL}}$ is left to check whether $U_i = P_{\underline{\mathsf{H}}_i}(\underline{r}_i)$. For $i < \ell$, the verifier performs the equivalent check $u_{i+1} = P_{\underline{\mathsf{h}}_{i+1}}(\underline{r}_i)$ in the next iteration of the loop. For $i = \ell$, the verifier $\mathbf{V}_{\mathsf{EVAL}}$ checks whether $U_\ell = P_{\underline{\mathsf{H}}_\ell}(\underline{r}_\ell)$ by computing $P_{\underline{\mathsf{H}}_\ell}(\underline{r}_\ell)$ directly.

Remark 5.3. The above construction has large soundness error because Lemma 4.3 only guarantees that for malicious $u_{i+1,j}, U_i, \kappa_{i+1}$

$$\Pr\left[\langle \underline{\rho}_{i+1}, (u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}})\rangle = \langle \underline{\rho}_{i+1}, \mathsf{up}_i(U_i) - \kappa_{i+1}\rangle\right] \leq 1/2.$$

We can reduce this soundness error of this step to $(1/2)^\mu$ as follows: In Item 3, the verifier sends μ challenges $\underline{\rho}_{i+1,1}, \ldots, \underline{\rho}_{i+1,\mu} \leftarrow \{0,1\}^{\delta_{i+1}}$ and the prover and verifier perform Item 4 and Item 5 for each of these challenges. If all the checks pass, then the verifier samples a random $j \in [\mu]$, and they continue to the next phase with respect to the challenge $\underline{\rho}_{i+1,j}$. The following theorem states the complexity measures for $\mu \in \mathbb{N}$.

Theorem 5.4. *Construction 5.2 satisfies the following properties:*

- ***Perfect completeness.***
- ***Soundness:*** *if* LM *is* $\log n$*-secure (Definition 4.6) and elimination-friendly (Definition 4.7), and for each* $i \in [\log n]$, $B_{\mathsf{BRA},i} \geq B_i^\star \cdot D_{\mathsf{LF},i}^{\log n - i + 1}$*, then it has soundness error* $2^{-\mu} + \frac{\log n \cdot (\log n - 1)}{|\mathcal{C}|} + \mathrm{negl}(\lambda)$.
- ***Communication:*** *for each* $i \in [\log n]$*, the prover sends* $O(\log n)$ *elements in* $M_{\mathrm{T},i}$, $O(\mu \cdot (\delta_i + \log n) \cdot \log n)$ *elements in* $M_{\mathrm{L},i}(B_i^\star)$*, and* $O(1)$ *elements in* $M_{\mathrm{R},i}$*, and for each* $i \in [\log n]$ *the verifier sends* $O(\log n)$ *elements of* $\mathcal{C}$ *and* $O(\mu \cdot \sum_{i=1}^{\log n} \delta_{i+1})$ *bits.*
- ***Prover efficiency:*** *the prover can be implemented in* $O(\mu \cdot (n/2^i + \delta_i))$ *operations in* $M_{\mathrm{R},i}$, $M_{\mathrm{T},i}$, $M_{\mathrm{L},i}(B_i^\star)$ *for each* $i \in [\log n]$*;* $O(\mu \cdot n)$ *applications of* e_i *for each* $i \in [\log n]$*;* $O(\mu)$ *applications of* up_i *for each* $i \in [\log n]$.
- ***Verifier efficiency:*** *the verifier performs* $O(\log^2 n)$ *operations in* R, $O(\mu \cdot (\log n - i + \delta_i))$ *operations in* $M_{\mathrm{T},i}$ *and* $M_{\mathrm{L},i}(\|\mathcal{C}\|_R \cdot B_i^\star)$ *for each* $i \in [\log n]$*; two norm-checks over* $M_{\mathrm{L},i}$ *and* $O(\mu)$ *applications of the mappings* e_i, up_i*, and* dn_i *for each* $i \in [\log n]$*; one operation in* $M_{\mathrm{R},\log n}$.

Above, $B_i^\star := 2^{\log n - i}\gamma_R^{\log n - i + 1}\|\mathcal{C}\|_R^{\log n - i}(1 + \gamma_R\|\mathcal{C}\|_R)\delta_i B_i$, $D_{\mathsf{LF},i} = 6\gamma_R^2 D_i^3 \iota(\mathcal{C}, 3)\|\mathcal{C}\|_R$ *and* $\mu \in \mathbb{N}$.

5.1 Efficiency

Communication. In the i-th phase, the prover sends $O(\log n - i)$ elements of $M_{\mathrm{T},i}$ and $M_{\mathrm{L},i}(B_i^\star)$, $O(1)$ elements of $M_{\mathrm{R},i}$, $O(\delta_{i+1})$ elements of $M_{\mathrm{L},i+1}(B_{i+1}^\star)$. In the i-th phase, the verifier sends $\log n - i$ elements of $\mathcal{C}$ and δ_{i+1} bits.

Prover efficiency. The prover performs $\ell := \log n$ proofs of polynomial evaluation for polynomials with $n/2^i$ coefficients for $i \in [\log n]$ in Item 1. In the i-th proof of polynomial evaluation, the prover efficiency is $O(2^{\ell - i})$ operations in $M_{\mathrm{R},i}$, $M_{\mathrm{T},i}$, and $M_{\mathrm{L},i}(B_i^\star)$, and $O(2^{\ell - i})$ applications of e_i. For the i-th step of Item 2, the prover requires two applications of up_i, $O(2^{\ell - i})$ operations in $M_{\mathrm{L},i+1}(B_i^\star)$. Finally, for the i-th step of Item 4, the prover performs $O(\delta_{i+1})$ operations in $M_{\mathrm{T},i}$ and $M_{\mathrm{L},i}(B_i^\star)$.

Verifier efficiency. Similarly to the prover, for implementing Item 1 the verifier complexity can be expressed in terms of the verifier complexity of ℓ proofs of polynomial evaluation for polynomials with differing number of coefficients. In the i-th proof of polynomial evaluation, the verifier efficiency is $O(\log n - i)$ operations in $M_{\mathrm{T},i}$ and $M_{\mathrm{L},i}(B_i^\star)$, and one norm-check over $M_{\mathrm{L},i}$. Additionally, for the i-th step of Item 5, the verifier evaluates $P_{\underline{\tilde{r}}_{i-1}}(\underline{r}_i)$ as $\langle \bigotimes_{j=1}^{\ell-i}(1, r_{i-1,j+1}), \bigotimes_{j=1}^{\ell-i}(1, r_{i,j})\rangle = \prod_{j=1}^{\ell-i}(1 + r_{i-1,j+1}r_{i,j})$, which requires $O(\log n - i)$ operations in $R(\|\mathcal{C}\|_R)$, performs $O(\delta_{i+1})$ operations in $M_{\mathrm{T},i}$ and $M_{\mathrm{L},i+1}(B^\star_{i+1})$, an application of the mappings e_i, up_i, and dn_{i+1}, a multiplication in $M_{\mathrm{L},i}(\|\mathcal{C}\|_R \cdot B^\star_i)$, and a norm check in $M_{\mathrm{L},i}$. Finally, in Item 6 the verifier performs a multiplication in $M_{\mathrm{R},\log n}$.

5.2 Completeness

Let $p_{\underline{\mathsf{h}}}(\underline{s}) = u$. Fix any choice of verifier challenges $(\underline{r}_1, \dots, \underline{r}_\ell) \in \mathcal{C}^{(\ell-1)\ell/2}$. We show that the (honest) prover makes the verifier accept. By Lemma 3.6, for each sumcheck protocol of Item 1,

$$\begin{aligned}
&\sum_{\underline{\omega} \in \{-1,1\}^\ell} p^{(i)}_{\mathrm{sc}}(\underline{\omega}) = 2^\ell \left(\langle \underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R], \underline{\mathsf{H}}_i \rangle, \langle \textstyle\bigotimes_{j=1}^{\ell-i}(1, r_{i-1,j+1}), \underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R] \rangle \right) \\
&= 2^\ell \left(\langle \underline{\mathsf{h}}_i[L], \underline{\mathsf{H}}_i \rangle + r_{i-1,1} \langle \underline{\mathsf{h}}_i[R], \underline{\mathsf{H}}_i \rangle, \langle \textstyle\bigotimes_{j=1}^{\ell-i+1}(1, r_{i-1,j}), (\underline{\mathsf{h}}_i[L], \underline{\mathsf{h}}_i[R]) \rangle \right) \\
&= 2^\ell (\mathsf{cmL}_i + r_{i-1,1}\mathsf{cmR}_i, P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1})) \ .
\end{aligned}$$

The honest prover sends

$$\begin{aligned}
&w_i := P_{\underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{r}_i) \in M_{\mathrm{L},i}, && U_i := P_{\underline{\mathsf{H}}_i}(\underline{r}_i) \in M_{\mathrm{R},i}, \\
&\kappa_{i+1} := \mathsf{up}_i(U_i) - (u_{i+1,1}, \dots, u_{i+1,\delta_{i+1}}) \in M^{\delta_{i+1}}_{\mathrm{L},i+1}, && u_{i+1,j} := P_{\underline{\mathsf{h}}_{i+1,j}}(\underline{r}_i) \in M_{\mathrm{L},i+1} \text{ for } j \in [\delta_{i+1}],
\end{aligned}$$

where $(\underline{\mathsf{h}}_{i+1,1}, \dots, \underline{\mathsf{h}}_{i+1,\delta_{i+1}}) := \mathsf{up}_i(\underline{\mathsf{H}}_i) \in M^{n_i}_{\mathrm{L},i+1}$.

From the $(i-1)$-th iteration, $u_i = P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1})$ (by definition $u_1 = u$), hence the completeness of the sumcheck protocol guarantees that the i-th sumcheck verifier in Item 1 does not reject and outputs $(\underline{r}_i, (v_{i,0}, v_{i,1}))$, where $v_{i,0} = P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{r}_i) P_{\underline{\mathsf{H}}_i}(\underline{r}_i) \in M_{\mathrm{T},i}$ and $v_{i,1} = P_{\underline{\tilde{r}}_{i-1}}(\underline{r}_i) P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{r}_i) \in M_{\mathrm{L},i}$. Also, the prover sets $\underline{\mathsf{h}}_{i+1} := \sum_{j=1}^{\delta_{i+1}} \rho_{i+1,j} \underline{\mathsf{h}}_{i+1,j} \in M^{n_i}_{\mathrm{L},i+1}$ and $u_{i+1} := \langle \underline{\rho}_{i+1}, (u_{i+1,1}, \dots, u_{i+1,\delta_{i+1}}) \rangle \in M_{\mathrm{L},i+1}$, so $u_{i+1} = P_{\underline{\mathsf{h}}_{i+1}}(\underline{r}_i)$.

The verifier's check $(v_{i,0}, v_{i,1}) = (w_i \cdot U_i, P_{\underline{\tilde{r}}_{i-1}}(\underline{r}_i) \cdot w_i)$ passes, because of the definition of (w_i, U_i, u_{i+1}) in Item 2. Also, we have assumed that $B_s = \|\mathcal{C}\|_R$. Hence, $\|\underline{\mathsf{h}}_i\|_{M_{\mathrm{L},i}} \leq \gamma_R \cdot \delta_i \cdot B_i$, $\|\underline{r}_i\|_R \leq \|\mathcal{C}\|_R$ for $i \geq 0$, and

$$\|w_i\|_{M_{\mathrm{L},i}} = \|P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{r}_i)\|_{M_{\mathrm{L},i}} \leq 2^{\ell-i} \gamma_R^{\ell-i+1} \|\mathcal{C}\|_R^{\ell-i} (1 + \gamma_R \|\mathcal{C}\|_R) \delta_i B_i,$$

by writing the polynomial $P_{\underline{\mathsf{h}}_i[L]+r_{i-1,1}\underline{\mathsf{h}}_i[R]}(\underline{r}_i)$ as a sum of coefficients multiplied by monomials, applying the triangle inequality, and then applying the ring augmentation factors of Definition 3.2 and Definition 3.2 to each term. This shows that the verifier norm check of Item 5 passes. Additionally, from the properties of $\underline{\mathcal{M}}$ and the definition of κ_{i+1}, it holds that $\kappa_{i+1} \in \ker \mathsf{dn}_{i+1}$ and $\langle \underline{\rho}_{i+1}, (u_{i+1,1}, \dots, u_{i+1,\delta_{i+1}}) \rangle = \langle \underline{\rho}_{i+1}, \mathsf{up}_i(U_i) - \kappa_{i+1} \rangle$.

Finally, in Item 6 the verifier's check $U_\ell = P_{\underline{\mathsf{H}}_\ell}(\underline{r}_\ell)$ passes, because of the definition of U_ℓ in the honest prover messages.

5.3 Soundness

We introduce some notation. For $i \in \{1, \dots, \ell\}$, let $\mathbf{P}^{(i)}$ and $\mathbf{V}^{(i)}$ be the parts of the prover and verifier algorithms that correspond to the i-th iteration of Item 1. In particular, $\mathbf{P}^{(i)}$ and $\mathbf{V}^{(i)}$ take as input $\mathbb{x}^{(i)}_{\mathrm{SC}} = (\underline{r}_{i-1}, u_i, \mathsf{cmL}_i, \mathsf{cmR}_i)$, the prover $\mathbf{P}^{(i)}$ additionally knows $\underline{\mathsf{h}}_i := (\underline{\mathsf{h}}_i[L], \underline{\mathsf{h}}_i[R])$ and $\underline{\mathsf{H}}_i$. At the end of the i-th iteration, $\mathbf{V}^{(i)}$ learns $(\underline{r}_i, (v_{i,0}, v_{i,1}), w_i, U_i)$, if the $\mathbf{V}^{(i)}$ checks pass (otherwise $\langle \mathbf{P}^{(i)}, \mathbf{V}^{(i)} \rangle = \perp$).

First, we prove the following claim for each of the $\ell := \log n$ iterations. Informally, the claim states that for each iteration, if we start with an invalid instance, then the instance of the next level will be invalid as well.

Claim 5.5. *Assuming that* LM *is* $\log n$*-secure and elimination-friendly,* $B_{\mathsf{BRA},i} \geq B_i^\star \cdot D_{\mathsf{LF},i}^{\log n - i+1}$*,* $\mathsf{pp} \leftarrow \mathbf{G}_{\mathsf{EVAL}}(1^\lambda, n)$*,* $\mathsf{cmL}_i := \langle \underline{\mathsf{h}}_i[L], \underline{\mathsf{H}}_i\rangle$*, and* $\mathsf{cmR}_i := \langle \underline{\mathsf{h}}_i[R], \underline{\mathsf{H}}_i\rangle$*, it holds that for every polynomial-size adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} \mathrm{tr} := (\underline{r}_i, (v_{i,0}, v_{i,1}), (w_i, U_i, u_{i+1})) \\ \wedge \quad u_i \neq P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1}) \\ \wedge \quad U_i = P_{\underline{\mathsf{H}}_i}(\underline{r}_i) \\ \wedge \quad (v_{i,0}, v_{i,1}) = (w_i \cdot U_i, P_{\tilde{\underline{r}}_{i-1}}(\underline{r}_i) \cdot w_i) \\ \|w_i\|_{M_{\mathrm{L},i}} \leq B_i^\star \end{array}\middle|\begin{array}{c} (\underline{r}_{i-1}, u_i, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b \overset{\mathrm{tr}}{\leftarrow} \langle \mathcal{A}(\mathsf{aux}, \mathbb{x}_{\mathrm{SC}}^{(i)}), \mathbf{V}^{(i)}(\mathsf{ivk}, \mathbb{x}_{\mathrm{SC}}^{(i)})\rangle \end{array}\right] \leq \epsilon_0(i) + \epsilon_1 \ ,$$

where $\epsilon_0(i)$ *is the error of witness-extended emulation of the proofs of polynomial evaluations of Construction 3.22 for polynomials with* $n/2^i$ *coefficients, and* ϵ_1 *is the probability of breaking the* $\log n$*-level bilinear relation assumption.*

Proof. In the i-th iteration the prover and the verifier perform a proof of polynomial evaluation (Construction 3.22), which has witness-extended emulation as stated in Theorem 3.21. However, this soundness guarantee is for a relaxed relation, which is an issue in our case. We remove this relaxation factors by noticing that in our construction there is no witness, so we can compute the messages of the honest prover. We show that a cheating prover deviating from the messages of the honest prover must break the bilinear relation assumption.

Since the proof of polynomial evaluations of Construction 3.22 for polynomials with $n/2^i$ coefficients has witness-extended emulation with knowledge error $\epsilon_0(i)$, there exists an expected polynomial-time algorithm $\mathcal{E}$ such that for every polynomial-size adversary $\mathcal{A}$, we have

$$\left|\Pr\left[\mathcal{A}(\mathsf{aux}, \mathrm{tr}) = 1 \middle| \begin{array}{c} \mathsf{pp}_{\mathrm{LF}} \leftarrow \mathbf{G}_{\mathrm{LF}}(1^\lambda, 2^{\ell-i}) \\ (\mathbb{x}_{\mathrm{SC}}^{(i)}, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b \overset{\mathrm{tr}}{\leftarrow} \langle \mathcal{A}(\mathsf{aux}), \mathbf{V}(\mathbb{x}_{\mathrm{SC}}^{(i)})\rangle \end{array}\right] - \Pr\left[\begin{array}{c} \mathcal{A}(\mathsf{aux}, \mathrm{tr}) = 1 \\ \wedge \quad \mathrm{tr} \text{ is accepting} \\ \wedge \quad (\mathbb{x}_{\mathrm{SC}}^{(i)}, \mathbb{w}_{\mathrm{SC}}^{(i)}) \in \mathcal{R}_{\mathrm{SCA}}(\xi_{\mathsf{LF},i}, B_i^\star \cdot D_{\mathsf{LF},i}^{\log n - i+1}) \end{array}\middle| \begin{array}{c} \mathsf{pp}_{\mathrm{LF}} \leftarrow \mathbf{G}_{\mathrm{LF}}(1^\lambda, 2^{\ell-i}) \\ (\mathbb{x}_{\mathrm{SC}}^{(i)}, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathrm{tr}, \mathbb{w}_{\mathrm{SC}}^{(i)}) \leftarrow \mathcal{E}^{\mathcal{A}(\mathsf{aux})}(\mathsf{pp}, \mathbb{x}_{\mathrm{SC}}^{(i)}) \end{array}\right]\right| \leq \epsilon_0(i) \ ,$$

where $\mathbf{G}_{\mathrm{LF}}(1^\lambda, 2^{\ell-i})$ outputs the parameters of $\mathbf{G}_{\mathsf{EVAL}}(1^\lambda, n)$ corresponding to the i-th level of $\underline{\mathcal{M}}$, $\mathbf{V}_{\mathsf{SA}}$ is the verifier of the sumcheck argument, and $\xi_{\mathsf{LF},i} := \xi_i^{3(\log n - i+1)}$.

If $(\mathbb{x}_{\mathrm{SC}}^{(i)}, \mathbb{w}_{\mathrm{SC}}^{(i)}) \in \mathcal{R}_{\mathrm{SCA}}(\xi_{\mathsf{LF},i}, B_i^\star \cdot D_{\mathsf{LF},i}^{\log n - i+1})$, it must be that

$$(\langle \mathbb{w}_{\mathrm{SC}}^{(i)}, \underline{\mathsf{H}}_i\rangle, \langle \tilde{\underline{r}}_{i-1}, \mathbb{w}_{\mathrm{SC}}^{(i)}\rangle) = (\xi_{\mathsf{LF},i} \cdot (\mathsf{cmL}_i + r_{i-1,1}\mathsf{cmR}_i), \xi_{\mathsf{LF},i} \cdot u_i)$$

and $\|\mathbb{w}_{\mathrm{SC}}^{(i)}\|_{M_{\mathrm{L},i}} \leq B_i^\star \cdot D_{\mathsf{LF},i}^{\log n}$. On the other hand, by assumption

$$\langle \underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R], \underline{\mathsf{H}}_i\rangle = \mathsf{cmL}_i + r_{i-1,1}\mathsf{cmR}_i \ .$$

If $\mathbb{w}_{\mathrm{SC}}^{(i)} \neq \xi_{\mathsf{LF},i} \cdot (\underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R])$, then $\mathbb{w}_{\mathrm{SC}}^{(i)} - \xi_{\mathsf{LF},i} \cdot (\underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R])$ breaks the leveled bilinear relation assumption (for Check that tests whether $B_i^\star \cdot D_{\mathsf{LF},i}^{\log n - i+1} + \|\xi_{\mathsf{LF},i}\|_R(1+\gamma_R\|\mathcal{C}\|_R)\gamma_R \cdot \delta_i \cdot B_i \leq B_{\mathsf{BRA},i}$)

since $\langle \mathbb{w}_{\mathrm{SC}}^{(i)} - \xi_{\mathsf{LF},i} \cdot (\underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R]), \underline{\mathsf{H}}_i\rangle = 0$. Hence, it must be that $\mathbb{w}_{\mathrm{SC}}^{(i)} = \xi_{\mathsf{LF},i} \cdot (\underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R])$ with $1 - \epsilon_1$ probability.

Finally, the evaluation $P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1})$ can be computed as $\langle \bigotimes_{j=1}^{\ell-i+1}(1, r_{i-1,j}), \underline{\mathsf{h}}_i\rangle$, and since $\underline{\tilde{r}}_{i-1} = \bigotimes_{j=1}^{\ell-i}(1, r_{i-1,j+1})$,

$$\begin{aligned}\xi_{\mathsf{LF},i} \cdot u_i &= \langle \underline{\tilde{r}}_{i-1}, \mathbb{w}_{\mathrm{SC}}^{(i)}\rangle = \langle \underline{\tilde{r}}_{i-1}, \xi_{\mathsf{LF},i}(\underline{\mathsf{h}}_i[L] + r_{i-1,1}\underline{\mathsf{h}}_i[R])\rangle \\ &= \langle \textstyle\bigotimes_{j=1}^{\ell-i+1}(1, r_{i-1,j}), \xi_{\mathsf{LF},i} \cdot \underline{\mathsf{h}}_i\rangle = \xi_{\mathsf{LF},i} \cdot P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1}) \ .\end{aligned}$$

This implies that $P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1}) = u_i$ because LM is elimination-friendly, and hence $\xi_{\mathsf{LF},i} := \xi_i^{3(\ell-i+1)}$ is not a zero-divisor. □

Using Claim 5.5, we show that Construction 5.2 has soundness error $\sum_{i=1}^{\ell} \epsilon_0(i) + \ell \cdot (\epsilon_1 + \epsilon_{\mathsf{LM}}(\mu))$, where $\epsilon_0(i)$ is the knowledge error of Construction 3.22, ϵ_1 is the probability of breaking the $\log n$-level bilinear relation assumption, and $\epsilon_{\mathsf{LM}}(\mu) = 2(1/2)^{\mu}$. From Theorem 3.21 and the fact that LM is ℓ-secure, $\epsilon_0(i) = \frac{\ell-i}{|\mathcal{C}|} + \mathrm{negl}(\lambda)$ and $\epsilon_1 = \mathrm{negl}(\lambda)$. We show that the total soundness error is $\ell \cdot \epsilon_{\mathsf{LM}}(\mu) + \frac{\ell(\ell-1)}{|\mathcal{C}|} + \mathrm{negl}(\lambda)$. Specifically, we prove by induction that the event $u_1 \neq P_{\underline{\mathsf{h}}_1}(\underline{r}_0)$ holds for $\tilde{\mathbf{P}}$ with probability at most $\sum_{i=1}^{\ell} \epsilon_0(i) + \ell \cdot (\epsilon_1 + \epsilon_{\mathsf{LM}}(\mu))$ over the verifier randomness $\underline{r}_1, \ldots, \underline{r}_\ell$.

Let $\tilde{\mathbf{P}}$ be a prover for Construction 5.2. For $i = \ell$, from the final verifier check $U_\ell = P_{\underline{\mathsf{H}}_\ell}(\underline{r}_\ell)$, so Claim 5.5 guarantees that $u_\ell \neq P_{\underline{\mathsf{h}}_\ell}(\underline{r}_{\ell-1})$ with probability at most $\epsilon_0(\ell) + \epsilon_1$ over the verifier randomness $\underline{r}_\ell$. Assume for the inductive hypothesis that $u_{i+1} \neq P_{\underline{\mathsf{h}}_{i+1}}(\underline{r}_i)$, where $\underline{\mathsf{h}}_{i+1} := \sum_{j=1}^{\delta_{i+1}} \rho_{i+1,j}\underline{\mathsf{h}}_{i+1,j}$, holds for $\tilde{\mathbf{P}}$ with probability at most $\sum_{j=\ell-i}^{\ell} \epsilon_0(j) + i \cdot \epsilon_1$ over the verifier randomness $\underline{r}_{i+1}, \ldots, \underline{r}_\ell$. We show that $U_i \neq P_{\underline{\mathsf{H}}_i}(\underline{r}_i)$ as follows

$$\begin{aligned}& u_{i+1} \neq P_{\underline{\mathsf{h}}_{i+1}}(\underline{r}_i) && \text{(from the inductive hypothesis)} \\ \implies\ & \langle \underline{\rho}_{i+1}, (u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}})\rangle \neq P_{\underline{\mathsf{h}}_{i+1}}(\underline{r}_i) && \text{(from verifier computation in Item 5)} \\ \implies\ & \langle \underline{\rho}_{i+1}, (u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}})\rangle \neq \langle \underline{\rho}_{i+1}, (P_{\underline{\mathsf{h}}_{i+1,1}}(\underline{r}_i), \ldots, P_{\underline{\mathsf{h}}_{i+1,\delta_{i+1}}}(\underline{r}_i))\rangle && \text{(from definition of } \underline{\mathsf{h}}_{i+1}\text{)} \ .\end{aligned}$$

If the above check is repeated μ times, then from Lemma 4.3, it must be that $(u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}}) \neq P_{\underline{\mathsf{h}}_{i+1,1}}(\underline{r}_i), \ldots, P_{\underline{\mathsf{h}}_{i+1,\delta_{i+1}}}(\underline{r}_i)$ with probability at most $\epsilon_{\mathsf{LM}}(\mu)/2$. Additionally, the verifier checks in Item 5 that $\langle \underline{\rho}_{i+1}, (u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}})\rangle = \langle \underline{\rho}_{i+1}, \mathsf{up}_i(U_i) - \kappa_{i+1}\rangle$. Again from the properties of $\underline{\mathcal{M}}$, it must be that $(u_{i+1,1}, \ldots, u_{i+1,\delta_{i+1}}) \neq \mathsf{up}_i(U_i) - \kappa_{i+1}$ with probability at most $\epsilon_{\mathsf{LM}}(\mu)/2$. Conditioning on the complements of these events,

$$(P_{\underline{\mathsf{h}}_{i+1,1}}(\underline{r}_i), \ldots, P_{\underline{\mathsf{h}}_{i+1,\delta_{i+1}}}(\underline{r}_i)) = \mathsf{up}_i(U_i) - \kappa_{i+1}.$$

The verifier checks that $\kappa_{i+1} \in \ker \mathsf{dn}_{i+1}$, so by applying dn_{i+1} we get that

$$P_{\underline{\mathsf{H}}_i}(\underline{r}_i) = U_i.$$

Then, from Claim 5.5 $u_i \neq P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1})$ with probability at most $\epsilon_0(i) + \epsilon_1$ over the verifier randomness $\underline{r}_i$. Hence, it must be that for $\tilde{\mathbf{P}}$, $u_i \neq P_{\underline{\mathsf{h}}_i}(\underline{r}_{i-1})$ with probability at most $\sum_{j=\ell-i+1}^{\ell} \epsilon_0(j) + (i+1) \cdot (\epsilon_1 + \epsilon_{\mathsf{LM}}(\mu))$ over the verifier randomness $\underline{r}_i, \ldots, \underline{r}_\ell$.

6 Sumcheck arguments with succinct verifier

Sumcheck arguments are succinct interactive arguments for openings of sumcheck-friendly commitment schemes (see Section 3.5). We use the delegation protocol from Section 5 to achieve, in the preprocessing setting, a succinct verifier for sumcheck arguments by delegating the key polynomial evaluation in Construction 3.18. For Theorem 6.2 below, we additionally need a compatibility property between the setup of the commiment scheme and leveled-bilinear modules.

Definition 6.1. *A sumcheck-friendly commitment scheme* $\mathsf{CM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Open})$ *is* **compatible** *with the preprocessing interactive argument* $\mathsf{EVAL} = (\mathbf{G}_{\mathsf{EVAL}}, \mathbf{I}_{\mathsf{EVAL}}, \mathbf{P}_{\mathsf{EVAL}}, \mathbf{V}_{\mathsf{EVAL}})$ *for* $\mathcal{R}_{\mathsf{EVAL}}$ *of Construction 5.2 if* $\mathsf{pp}_{\mathsf{CM}} \in \mathsf{CM}.\mathsf{Setup}(1^\lambda, n)$ *contains* $(\underline{\mathcal{M}}, \mathsf{aux}) \leftarrow \mathbf{G}_{\mathsf{EVAL}}(1^\lambda, 2n)$*, and* $\mathbb{K} := M^n_{\mathrm{R},1}$.

Our construction is a succinct argument for $\mathcal{R}_{\mathrm{SCA}}$ from Definition 3.17.

Theorem 6.2. *There is a public-coin preprocessing interactive argument* $\mathsf{SCA} = (\mathbf{G}_{\mathrm{SCA}}, \mathbf{I}_{\mathrm{SCA}}, \mathbf{P}_{\mathrm{SCA}}, \mathbf{V}_{\mathrm{SCA}})$ *with the following properties:*

- *Perfect completeness for* $\mathcal{R}_{\mathrm{SCA}}(1, B_{\mathsf{c}})$.
- *Witness-extended emulation for the relation* $\mathcal{R}_{\mathrm{SCA}}(\xi^\ell, B_{\mathcal{E}})$*, where* $\ell := \log n$ *and* $B_{\mathcal{E}} := D^\ell \cdot (B_{\mathsf{c}} \cdot (d_{\mathsf{ck}} + 1)^\ell \, \gamma_R^{\ell d_{\mathsf{ck}}} \, \|\mathcal{C}\|_R^{\ell d_{\mathsf{ck}}})$*, with knowledge error* $\epsilon_0 + \epsilon_1$*, where* ϵ_0 *is the soundness error of* EVAL *and* ϵ_1 *is the knowledge error of the sumcheck argument for* CM *if*
  - CM *is* $(K, B_{\mathsf{INV}}, D, \xi)$*-invertible;*
  - CM *is compatible with* EVAL *according to Definition 6.1;*
  - *the conditions of Theorem 5.4 hold.*
- *Communication complexity is the sum of the communication cost of sumcheck argument for* CM *and the delegation protocol* EVAL.
- *Prover (resp. verifier) complexity is equal to the sum of the prover (resp. verifier) complexity of the sumcheck argument for* CM *(without the last verifier polynomial evaluation) and the prover (resp. verifier) complexity of* EVAL.

We state Construction 6.3 and then in Section 6.1 we show that it realizes Theorem 6.2.

Construction 6.3. We describe a public-coin interactive argument $\mathsf{SCA} = (\mathbf{G}_{\mathrm{SCA}}, \mathbf{I}_{\mathrm{SCA}}, \mathbf{P}_{\mathrm{SCA}}, \mathbf{V}_{\mathrm{SCA}})$ for the relation $\mathcal{R}_{\mathrm{SCA}}$, which uses a delegation protocol (with preprocessing) for multilinear polynomial evaluation $\mathsf{EVAL} = (\mathbf{G}_{\mathsf{EVAL}}, \mathbf{I}_{\mathsf{EVAL}}, \mathbf{P}_{\mathsf{EVAL}}, \mathbf{V}_{\mathsf{EVAL}})$.

Generator. Let $\mathbf{G}_{\mathrm{SCA}}(1^\lambda, n)$ be an algorithm that runs $\mathsf{pp}_{\mathsf{CM}} \leftarrow \mathsf{CM}.\mathsf{Setup}(1^\lambda, n)$, which as a first step samples $\mathsf{pp}_{\mathsf{EVAL}} \leftarrow \mathbf{G}_{\mathsf{EVAL}}(1^\lambda, 2n)$, $\mathsf{ck} \leftarrow \mathsf{CM}.\mathsf{KeyGen}(\mathsf{pp}_{\mathsf{CM}}, B_{\mathsf{ck}})$, and outputs $\mathsf{pp} = (\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck})$.

Indexer. Let $\mathbf{I}_{\mathrm{SCA}}(\mathsf{pp}, \mathbb{i})$ be an algorithm that samples $\underline{\rho} \leftarrow \{0,1\}^{\delta_1}$ and sets

$$\underline{\mathcal{M}}' := \Big((\mathcal{M}_i)_{i=2}^{\log n+1}, (B_i)_{i=2}^{\log n+1}, (\mathsf{up}_i, \mathsf{dn}_i, \delta_{i+1})_{i=2}^{\log n} \Big),$$

$\mathsf{aux}' := ((B_{\mathsf{BRA},i}, \xi_i, D_i)_{i=2}^{\log n+1}, \mathcal{C})$ and $\mathsf{pp}'_{\mathsf{EVAL}} := (\underline{\mathcal{M}}', \mathsf{aux}')$, runs $(\mathsf{ipk}_{\mathsf{EVAL}}, \mathsf{ivk}_{\mathsf{EVAL}}) = \mathbf{I}_{\mathsf{EVAL}}(\mathsf{pp}'_{\mathsf{EVAL}}, \mathbb{i}_{\mathsf{EVAL}})$, where $\mathbb{i}_{\mathsf{EVAL}} = \langle \underline{\rho}, \mathsf{up}_1(\mathsf{ck}) \rangle$, and outputs $(\mathsf{ipk}, \mathsf{ivk}) = ((\mathsf{pp}_{\mathsf{CM}}, \underline{\rho}, \mathsf{ipk}_{\mathsf{EVAL}}), (\mathsf{pp}_{\mathsf{CM}}, \underline{\rho}, \mathsf{ivk}_{\mathsf{EVAL}}))$.

The interactive phase. The prover $\mathbf{P}_{\mathrm{SCA}}$ takes $(\mathsf{ipk}, \mathbb{x}, \mathbb{w})$ as input and the verifier $\mathbf{V}_{\mathrm{SCA}}$ takes as input $(\mathsf{ivk}, \mathbb{x})$. The protocol proceeds as follows.

1. The prover $\mathbf{P}_{\mathrm{SCA}}$ and the verifier $\mathbf{V}_{\mathrm{SCA}}$ run a sumcheck argument $\mathsf{SCA}' = (\mathbf{P}', \mathbf{V}')$ (without the verifier key polynomial evaluation) as in Construction 3.18.

Protocol: SCA'

The prover $\mathbf{P}'$ and verifier $\mathbf{V}'$ take as input an instance $\mathbb{x} = (\mathsf{CM}, \mathsf{pp}_{\mathsf{CM}}, \mathcal{C}, \mathsf{ck}, \mathsf{cm})$; the prover $\mathbf{P}'$ additionally takes as input a witness $\mathbb{w} = (\mathsf{m}, \rho)$. Here, CM is sumcheck-friendly with respect to the ring R and subset $\mathcal{H}$.

The prover $\mathbf{P}'$ and verifier $\mathbf{V}'$ engage in a sumcheck protocol (described in Protocol 1) for the instance

$$\mathbb{x}_{\mathrm{SC}} := \Big(R, M = \mathbb{C}, \mathcal{H}, \ell = \log n, \tau = \mathsf{cm}, \mathcal{C}\Big)$$

where the prover $\mathbf{P}'$ uses the polynomial $p_{\mathsf{sc}}(\underline{X}) := f_{\mathsf{CM}}(P_{\mathsf{m}}(\underline{X}), P_{\mathsf{ck}}(\underline{X}), 1)$ induced by $\mathbb{x}$ and $\mathbb{w}$. After the end of the sumcheck protocol, the prover $\mathbf{P}'$ learns $\underline{r} \in \mathcal{C}^{\ell}$ and the verifier $\mathbf{V}'$ learns $(\underline{r}, v) \in \mathcal{C}^{\ell} \times \mathbb{C}$. (If the sumcheck verifier rejects, then $\mathbf{V}'$ rejects.) The prover $\mathbf{P}'$ computes and sends $w := P_{\mathsf{m}}(\underline{r}) \in \mathbb{M}$ and $U := P_{\mathsf{ck}}(\underline{r}) \in \mathbb{K}$ to the verifier $\mathbf{V}'$. ~~The verifier $\mathbf{V}'$ computes $P_{\mathsf{ck}}(\underline{r}) \in \mathbb{K}$~~
The verifier $\mathbf{V}'$ checks that $\|w\|_{\mathbb{M}} \leq B_{\mathsf{c}} \cdot (d_{\mathsf{ck}} + 1)^{\ell}\, \gamma_R^{\ell d_{\mathsf{ck}}}\, \|\mathcal{C}\|_R^{\ell d_{\mathsf{ck}}}$, checks that $f_{\mathsf{CM}}(w, U, 1) = v$, and checks that $\alpha_{\mathsf{sc}}(\mathsf{ck}, w, \underline{r}) = 1$.

2. The prover $\mathbf{P}_{\mathrm{SCA}}$ computes

$$\begin{aligned}
&(\underline{\mathsf{h}}_1, \ldots, \underline{\mathsf{h}}_{\delta_2}) := \mathsf{up}_1(\mathsf{ck}) \in (M^n_{\mathrm{L},2}(B_2))^{\delta_2} \\
&\underline{\mathsf{h}} := \langle \rho, \mathsf{up}_1(\mathsf{ck}) \rangle \in M^n_{\mathrm{L},2}(B_2) \\
&u_j := P_{\underline{\mathsf{h}}_j}(\underline{r}) \in M_{\mathrm{L},2} \text{ for } j \in [\delta_2] \\
&u := P_{\underline{\mathsf{h}}}(\underline{r}) \in M_{\mathrm{L},2} \\
&\kappa := \mathsf{up}_1(U) - (u_1, \ldots, u_{\delta_2}) \in M^{\delta_2}_{\mathrm{L},2} \quad ,
\end{aligned}$$

   and sends $((\underline{\mathsf{h}}_1, \ldots, \underline{\mathsf{h}}_{\delta_2}), (u_1, \ldots, u_{\delta_2}), \kappa)$.

3. The verifier $\mathbf{V}_{\mathrm{SCA}}$ computes $u := \langle \underline{\rho}, (u_1, \ldots, u_{\delta_2}) \rangle \in M_{\mathrm{L},2}$, and checks that

$$\begin{aligned}
&\langle \underline{\rho}, (u_1, \ldots, u_{\delta_2}) \rangle = \langle \underline{\rho}, \mathsf{up}_1(u) - \kappa \rangle \quad , \\
&\kappa \in \ker \mathsf{dn}_1 \quad .
\end{aligned}$$

4. The prover $\mathbf{P}_{\mathrm{SCA}}$ and the verifier $\mathbf{V}_{\mathrm{SCA}}$ run $\mathbf{P}_{\mathsf{EVAL}}(\mathsf{ipk}_{\mathsf{EVAL}}, \mathbb{x}_{\mathsf{EVAL}}, \mathbb{w}_{\mathsf{EVAL}})$ and $\mathbf{V}_{\mathsf{EVAL}}(\mathsf{ivk}_{\mathsf{EVAL}}, \mathbb{x}_{\mathsf{EVAL}})$ from Construction 5.2 to prove that $u = P_{\underline{\mathsf{h}}}(\underline{r})$ for

$$(\mathsf{pp}'_{\mathsf{EVAL}}, \mathbb{i}_{\mathsf{EVAL}}, \mathbb{x}_{\mathsf{EVAL}}, \mathbb{w}_{\mathsf{EVAL}}) := \Big(\mathsf{pp}'_{\mathsf{EVAL}}, \underline{\mathsf{h}}, (\|\mathcal{C}\|_R, \underline{r}, u), \emptyset\Big) \quad .$$

6.1 Proof of Theorem 6.2

Completeness, efficiency and communication follow directly from the corresponding properties of sumcheck arguments and EVAL.

For the soundness, we combine the witness-extended emulation of SCA' (Theorem 3.19) with the soundness of EVAL (Theorem 5.4). Informally, the soundness of EVAL guarantees that if $U \neq P_{\mathsf{ck}}(\underline{r})$, then the verifier accepts with probability at most ϵ_0, and the guarantee of SCA' states that the protocol has witness-extended emulation with error ϵ_1 given that $U = P_{\mathsf{ck}}(\underline{r})$. Combining the two guarantees, we show that SCA has witness-extended emulation with error $\epsilon_0 + \epsilon_1$.

A subtle point in the proof is that the properties of the subprotocols hold with respect to the randomness of $\mathsf{CM.Setup}(1^\lambda, n)$ and $\mathbf{V}'$, and the randomness of $\mathbf{G}_{\mathsf{EVAL}}(1^\lambda, 2n)$ and $\mathbf{V}_{\mathsf{EVAL}}$, respectively. However, in SCA, $\mathsf{CM.Setup}(1^\lambda, n)$ and $\mathbf{G}_{\mathsf{EVAL}}(1^\lambda, 2n)$ are not run independently. To circumvent this issue, we notice that the sequential composition has a special form that allows any adversary that breaks SCA with probability more than $\epsilon_0 + \epsilon_1$ to be transformed directly into an adversary of EVAL with success probability more than ϵ_0 or of SCA' with success probability more than ϵ_1 by simulating the other subprotocol.

7 Holographic polynomial IOPs over product rings

We construct holographic polynomial IOPs for the R1CS problem over a class of product rings, which includes the cyclotomic rings used in many lattice-based cryptosystems. This is a necessary building block for succinct, lattice-based arguments for R1CS as existing polynomial IOPs are defined over finite fields. Our construction extends a holographic IOP for R1CS over finite fields to product rings.

Definition 7.1 (R1CS). *The indexed relation $\mathcal{R}_{\mathrm{R1CS}}$ is the set of all triples*

$$(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \mathbb{w}) = \big(R_\bullet, (M, n_{\mathrm{row}}, n_{\mathrm{col}}, A, B, C), (n_{\mathrm{in}}, \underline{x}), \underline{w}\big)$$

where $R_\bullet$ is a ring, A, B, C are matrices in $R_\bullet^{n_{\mathrm{row}} \times n_{\mathrm{col}}}$, each with at most M non-zero entries, $\underline{x} \in R_\bullet^{n_{\mathrm{in}}}$, $\underline{w} \in R_\bullet^{n_{\mathrm{col}}-n_{\mathrm{in}}}$, and $\underline{z} := (\underline{x}, \underline{w}) \in R_\bullet^{n_{\mathrm{col}}}$ is a vector such that $A\underline{z} \circ B\underline{z} = C\underline{z}$. (Here "$\circ$" denotes the entry-wise product between two vectors.)

Theorem 7.2. *For every ring $R_\bullet$ such that $R_\bullet \simeq \mathbb{F}^k$, and positive integer ℓ, there is a holographic polynomial IOP over ring $R_\bullet$, with non-adaptive queries, for the indexed relation $\mathcal{R}_{\mathrm{R1CS}}$ that supports instances over $R_\bullet$ with $M = m \cdot 2^\ell$, $n_{\mathrm{row}} = 2^\ell$,$n_{\mathrm{col}} = 2^\ell$ and $n_{\mathrm{in}} = 2^{t_{\mathrm{in}}}$, and has the following properties (with $N = \max\{n_{\mathrm{row}}, n_{\mathrm{col}}\}$):*
- *the soundness error is $O((M+N)/|\mathbb{F}|)$;*
- *the proof length is $O(M+N)$ elements in $R_\bullet$, consisting of $O(m)$ multilinear polynomials;*
- *the round complexity is $O(\log(M+N))$;*
- *the query complexity is $O(1)$;*
- *the prover sends $O(\log(M+N))$ non-oracle messages in $R_\bullet$;*
- *the indexer uses $O(M)$ operations in $R_\bullet$;*
- *the prover uses $O(N+M)$ operations in $R_\bullet$;*
- *the verifier uses $O(n_{\mathrm{in}} + m + \log M)$ operations in $R_\bullet$.*

Using Lemma 7.3, it is easy to see that Theorem 7.2 holds for cyclotomic rings of the form $R_\bullet := \mathbb{Z}_p[X]/\langle \Phi_d(X) \rangle$ for a prime p, which are used in cryptographic systems based on ideal lattices. Thus, Theorem 7.2 can be used to construct proof systems about lattice-based cryptosystems.

Lemma 7.3. *Let $d \in \mathbb{N}$ and let p be a prime which does not divide d and has order t in $(\mathbb{Z}/d\mathbb{Z})^\times$. Let $\Phi_d(X)$ be the d-th cyclotomic polynomial. Then the factors of $\Phi_d(X)$ modulo p are distinct and each has degree equal to the order of p in $(\mathbb{Z}/d\mathbb{Z})^\times$. Moreover, the ring $\mathbb{Z}_p[X]/(\Phi_d(X))$ is isomorphic to $\mathbb{F}_{p^t}^{\phi(d)/t}$.*

Lemma 7.3 follows from [Con13, Theorem 5.3] using the Chinese Remainder Theorem and standard facts about finite fields.

7.1 Proof of Theorem 7.2

We prove Theorem 7.2 by mapping an R1CS instance over $R_\bullet$ to k instances over $\mathbb{F}$, and then using a result on IOPs from [BCG20]. Since $R_\bullet \simeq \mathbb{F}^k$, applying the isomorphism to each element of an R1CS instance over $R_\bullet$ maps the instance to k R1CS instances over $\mathbb{F}$. The non-zero entries of the R1CS matrices from these k R1CS instances over $\mathbb{F}$ are subsets of the non-zero entries of the $R_\bullet$-instance. Conversely, applying the isomorphism in reverse, using e.g. the k (I, J)-th entries of matrices $(A_i)_{i \in [k]}$ over $\mathbb{F}$ to produce the (I, J)-th entry of a matrix over $R_\bullet$ gives an R1CS instance over $R_\bullet$.

Theorem 7.4 ([BCG20, Theorem 6.1]). *For every finite field $\mathbb{F}$ and positive integer ℓ, there is a holographic polynomial IOP over ring $\mathbb{F}$, with non-adaptive queries, for the indexed relation $\mathcal{R}_{\mathrm{R1CS}}$ that supports instances over $\mathbb{F}$ with $M = m \cdot 2^{\ell}$, $n_{\mathrm{row}} = 2^{\ell}$,$n_{\mathrm{col}} = 2^{\ell}$ and $n_{\mathrm{in}} = 2^{t_{\mathrm{in}}}$, and has the following properties (with $N = \max\{n_{\mathrm{row}}, n_{\mathrm{col}}\}$):*

- *the soundness error is $O((M+N)/|\mathbb{F}|)$;*
- *the proof length is $O(M+N)$ elements in $\mathbb{F}$, consisting of $O(m)$ polynomials of degree at most $O(2^{\ell})$;*
- *the round complexity is $O(\log(M+N))$;*
- *the query complexity is $O(1)$;*
- *the prover sends $O(\log(M+N))$ non-oracle messages in $\mathbb{F}$;*
- *the indexer uses $O(M)$ field operations;*
- *the prover uses $O(M+N)$ field operations;*
- *the verifier uses $O(n_{\mathrm{in}} + m + \log N)$ field operations.*

Theorem 7.4 was originally stated as a tensor query IOP, but immediately gives a polynomial IOP for multilinear polynomials, since every tensor query in the [BCG20] construction can be rewritten as a multilinear polynomial evaluation.

Theorem 7.2 follows by executing the polynomial IOP of Theorem 7.4 k times in a vectorized, SIMD fashion over $\mathbb{F}^k$ with the same verifier randomness (over $\mathbb{F}$) for each $i \in [k]$. This gives k polynomial IOP executions over $\mathbb{F}$ which can then be mapped to a single polynomial IOP over $R_\bullet$ via the isomorphism between $R_\bullet$ and $\mathbb{F}^k$.

This is possible because the prover in Theorem 7.4 is the same in every copy of $\mathbb{F}$, since most of the polynomial IOP construction of [BCG20] can be modelled using arithmetic circuits over $\mathbb{F}$. In a few places in the construction proving Theorem 7.4, a linear scan, which is a non-algebraic operation, is applied to the sparse representations of R1CS matrices over $\mathbb{F}$. The structure of the result depends only on the positions of the non-zero entries in the R1CS instance, and not the witness. We can always include some zero entries in the sparse representations of the k R1CS instances over $\mathbb{F}$ so that all of them have the same support as the original R1CS instance over $R_\bullet$. The prover algorithm will then perform the same operations in each of the k copies of the proof.

The round complexity is directly inherited from Theorem 7.4. The alphabet changes to $R_\bullet$, so that the proof length, query complexity, indexer, prover and verifier complexity are the same but measured over $R_\bullet$ instead of $\mathbb{F}$. To determine the soundness error, note that given witnesses for the k instances over $\mathbb{F}$, we could apply the isomorphism to get a witness over $R_\bullet$. Conversely, if the R1CS instance over $R_\bullet$ is unsatisfiable, then there must be at least one derived R1CS instance over $\mathbb{F}$ which is unsatisfiable, in which case, the verifier will reject with probability $O((M+N)/|\mathbb{F}|)$ by Theorem 7.4. Therefore, the soundness error over $R_\bullet$ is also equal to $O((M+N)/|\mathbb{F}|)$ and does not depend on k.

8 Succinct verification for NP

We construct arguments for the R1CS problem with succinct verifier. Our main building blocks are a polynomial IOP for R1CS instances over product rings (Section 7) and a polynomial commitment scheme over rings with succinct verification (Section 8.1). In Section 8.2 we prove the following theorem.

Theorem 8.1 (formal restatement of Theorem 2)**.** *Let* $B_i^\star := 2^{\ell-i}\gamma_R^{\ell-i+1}\|\mathcal{C}\|_R^{\ell-i}(1+\gamma_R\|\mathcal{C}\|_R)\delta_i\cdot B_i$, $D_{\mathsf{LF},i} := 6\gamma_R^2 D_i^3\iota(\mathcal{C},3)\|\mathcal{C}\|_R$ *and* $\mu\in\mathbb{N}$. *Let* $\mathsf{LM} = (\mathsf{LM.Setup}, \mathsf{LM.KeyGen})$ *be a leveled bilinear-module generator for which it holds that* $M_{\mathrm{L},1}$ *is a ring such that* $M_{\mathrm{L},1}/I_1 \cong \mathbb{F}_p^k$ *for some* $k\in\mathbb{N}$ *and prime power* $p\in\mathbb{N}$.

There is a preprocessing argument of knowledge for R1CS over $R_\bullet := M_{\mathrm{L},1}/I_1$ *with* $M = m\cdot 2^\ell$, $n_{\mathrm{row}} = 2^\ell$, $n_{\mathrm{col}} = 2^\ell$ *and* $n_{\mathrm{in}} = 2^{t_{\mathrm{in}}}$*, and has the following properties (with* $N = \max\{n_{\mathrm{row}}, n_{\mathrm{col}}\}$*):*

- ***Perfect completeness;***
- ***Witness-extended emulation:*** *if* LM *is* ℓ*-secure, elimination-friendly, and quotient-friendly, and for each level* i*, it holds that* $B_{\mathsf{BRA},i} \geq B_i^\star \cdot D_{\mathsf{LF},i}^{\log n-i+1}$*, the argument has knowledge-soundness error* $O(2^{-\mu} + \frac{\log^2(M+N)}{|\mathcal{C}|} + \frac{M+N}{p}) + \mathrm{negl}(\lambda)$ *for the relation* $\mathcal{R}_{\mathrm{R1CS}}$.
- ***Communication complexity:*** $O(\mu\cdot\log^2(M+N))$ *elements in* $\mathcal{C}$, $M_{\mathrm{R},i}$, $M_{\mathrm{T},i}$, $M_{\mathrm{L},i}(B_i^\star)$*, and* $M_{\mathrm{L},i+1}(B_{i+1})$ *for* $i\in[\log N+1]$*;*
- ***Round complexity:*** $O(\log^2(M+N))$*;*
- ***Prover complexity:*** $O(M+\mu\cdot N)$ *operations in* $M_{\mathrm{R},i}$, $M_{\mathrm{T},i}$, $M_{\mathrm{L},i}(B_i^\star)$ *and applications of* e_i *and* up_i *for* $i\in[\log N+1]$*;*
- ***Verifier complexity:*** $O(n_{\mathrm{in}} + \mu\cdot\log^2(M+N))$ *operations in* $M_{\mathrm{T},i}$, $M_{\mathrm{L},i}(\|\mathcal{C}\|_R\cdot B_i^\star)$*, one norm-check over* $M_{\mathrm{L},i}$ *and an application of the mappings* up_i *and* dn_i *for each* $i\in[\log M+1]$*, and one operation in* $M_{\mathrm{R},\log N+1}$.

Specializing to our lattice-based instantiation of leveled bilinear-modules gives the following corollary.

Corollary 8.2 (formal restatement of Theorem 1)**.** *Let* $\mu\in\mathbb{N}$, $R := \mathbb{Z}[X]/\langle\Phi_d(X)\rangle$ *for* d *a prime power,* p *a prime which does not divide* t *and has order* t *in* $(\mathbb{Z}/d\mathbb{Z})^\times$*, and* $q > p$*. Assuming hardness of the SIS problem over* R/qR*, there is a preprocessing argument of knowledge with transparent setup for R1CS over* $R_\bullet := R/pR$ *with instances of size* $N = 2^\ell$ *with* $M = m\cdot 2^\ell$ *and* $n_{\mathrm{in}} = 2^{t_{\mathrm{in}}}$*, which has knowledge-soundness error* $O(2^{-\mu} + \frac{\log^2 N}{d} + \frac{N}{p^t})$*, round complexity* $O(\log^2 N)$*, communication complexity dominated by* $O(\mu\cdot\log^2(M+N))$ *elements of* R/qR*, prover complexity dominated by* $O(M+\mu\cdot N)$ *operations in* R/qR*, and verifier complexity dominated by* $O(n_{\mathrm{in}} + \mu\cdot\log^2(M+N))$ *operations in* R/qR.

Proof. Following Section 4.1, there is a leveled bilinear-module for $R = \mathbb{Z}[X]/\langle\Phi_d(X)\rangle$ for d a prime power. Set $I_i = pR$. By Lemma 7.3, $R_\bullet := R/pR$ is isomorphic to $\mathbb{F}_{p^t}^{\phi(d)/t}$ and satisfies the conditions of Theorem 8.1. The result follows. □

Specializing to the instantiation of leveled bilinear-modules based on bilinear groups gives the following corollary.

Corollary 8.3. *Let p be a prime, $R := \mathbb{F}$ the finite field with p elements, $\mathbb{G}_0$, $\mathbb{G}_1$ and $\mathbb{G}_{\mathrm{T}}$ be groups of order p, and $e\colon \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_{\mathrm{T}}$ be a bilinear map. Assuming the hardness of SXDH, there is a preprocessing argument of knowledge with transparent setup for R1CS over $R_\bullet := \mathbb{F}$ with instances of size $N = 2^\ell$ with $M = m \cdot 2^\ell$ and $n_{\mathrm{in}} = 2^{t_{\mathrm{in}}}$, which has knowledge-soundness error $O(\frac{\log^2 N}{p} + \frac{N}{p})$, round complexity $O(\log^2 N)$, communication complexity dominated by $O(\log^2(M+N))$ elements of $\mathbb{F}$, prover complexity dominated by $O(M+N)$ operations in $\mathbb{F}$, and verifier complexity dominated by $O(n_{\mathrm{in}} + \log^2(M+N))$ operations in $\mathbb{F}$.*

Proof. Following Section 4.1, $e, \mathbb{G}_0, \mathbb{G}_1$ and $\mathbb{G}_{\mathrm{T}}$ give a leveled bilinear-module for $R = \mathbb{F}$. Set $I_i = \{0\}$ so that $R_\bullet := R/\{0\} = \mathbb{F}$ and satisfies the conditions of Theorem 8.1. The result follows. $\square$

8.1 Polynomial commitments

We modify the commitment scheme described in Construction 3.22 to produce a polynomial commitment scheme whose evaluation algorithm has a succinct verifier.

Construction 8.4. Let $\mathsf{LM} = (\mathsf{LM.Setup}, \mathsf{LM.KeyGen})$ be a leveled bilinear-module generator. The polynomial commitment PC is defined via the following algorithms.

- $\mathsf{PC.Setup}(1^\lambda, n)$*:* sample $(\underline{\mathcal{M}}, \mathsf{aux}) \leftarrow \mathsf{LM.Setup}(1^\lambda, \log n + 1, 2n)$ and output $\mathsf{pp}_{\mathsf{CM}} := (\underline{\mathcal{M}}, \mathsf{aux})$.
- $\mathsf{PC.KeyGen}(\mathsf{pp}_{\mathsf{CM}}, n)$*:* sample $\mathsf{ck} \leftarrow \mathsf{LM.KeyGen}(\underline{\mathcal{M}}, \underline{n}, \mathsf{aux})$ with $n_1 = n$ and $n_i = 0$ for $i > 1$, and output $\mathsf{ck} \in M_{\mathrm{R},1}^n$.
- $\mathsf{PC.Commit}(\mathsf{ck}, \underline{P})$*:* given $\underline{P} \in (M_{\mathrm{L},1}/I_1)^n$, compute a representative $\underline{\boldsymbol{P}} \in M_{\mathrm{L},1}^n(B_1)$ of $\underline{P}$, such that $\underline{P} = \underline{\boldsymbol{P}} \bmod I_1$, and output $\mathsf{cm} := \langle \underline{\boldsymbol{P}}, \mathsf{ck} \rangle$.
- $\mathsf{PC.Open}(\mathsf{ck}, \underline{P}, \mathsf{cm}, \underline{\boldsymbol{P}}, c)$*:* check that $\mathsf{ck} \in M_{\mathrm{R}}^n$, $\underline{P} \in (M_{\mathrm{L},1}/I_1)^n$, $\underline{\boldsymbol{P}} \in M_{\mathrm{L},1}^n(B_{\mathsf{BRA},1})$, and $c \in R$ such that $c \cdot \underline{P} = \underline{\boldsymbol{P}} \bmod I_1$ and $c \cdot \mathsf{cm} = \langle \underline{\boldsymbol{P}}, \mathsf{ck} \rangle$.
- The interactive argument for $\mathcal{R}_{\mathrm{PC}}$, $\mathsf{PC\text{-}Eval} = (\mathbf{G}, \mathbf{I}, \mathbf{P}, \mathbf{V})$ is as follows:
  - $\mathbf{G}$ outputs $\mathsf{pp}_{\mathsf{CM}} \leftarrow \mathsf{PC.Setup}(1^\lambda)$;
  - $\mathbf{I}$ outputs $\perp$;
  - $\mathbf{P}$ and $\mathbf{V}$ run a sumcheck argument with succinct verifier (Construction 6.3) for the linear-function commitment $(\bigotimes_{i=1}^{\log n}(1, z_i), \mathsf{cm}, v)$, where the short description of $\ll \bigotimes_{i=1}^{\log n}(1, z_i) \gg = (z_1, \ldots, z_{\log n})$, to show that $\mathsf{cm} = \langle \underline{\boldsymbol{P}}, \mathsf{ck} \rangle$ and $v = \langle \bigotimes_{i=1}^{\log n}(1, z_i), \underline{P} \rangle := P(z_1, \ldots, z_{\log n}) \bmod I_1$.

Theorem 8.5. *Assume that* LM *is* $(\log n + 1)$*-secure,* PC *is a computationally binding commitment scheme such that*

- *the commitment key size is n elements of $M_{\mathrm{R},1}$;*
- *computing* $\mathsf{PC.Commit}$ *requires n applications of e_1 and $O(n)$ operations in $M_{\mathrm{L},1}$ and $M_{\mathrm{T},1}$.*

The preprocessing argument $\mathsf{PC\text{-}Eval}$ *for* PC *satisfies the following properties:*

- *It has perfect completeness*
- *It has witness-extended emulation with knowledge error $O(\log^2 n/|\mathcal{C}|)$ if* LM *satisfies the conditions of Theorem 5.4.*
- *Its communication complexity is the sum of the communication cost of sumcheck argument for* LF *(Theorem 3.21) and the delegation protocol* EVAL *(Theorem 5.4);*

- *Its prover (resp. verifier) complexity is equal to the sum of the prover (resp. verifier) complexity of the sumcheck argument for* LF *(without the last verifier polynomial evaluation) and the prover (resp. verifier) complexity of* EVAL*;*

Proof. The binding property of the commitment scheme follows immediately from the security of LM. The other properties follow from Theorem 6.2 instantiated with the parameters for sumcheck arguments for LF (Theorem 3.21). □

8.2 Sketch proof of Theorem 8.1

Theorem 7.2 gives a polynomial IOP for R1CS instances over $R_\bullet$. This polynomial IOP can be compiled into a preprocessing argument using the polynomial commitment scheme as follows:

- the setup algorithm consists of the setup algorithm for the polynomial commitment scheme;
- the indexer algorithm runs the indexer algorithms for the polynomial commitment scheme and the polynomial IOP, and computes commitments to each of the oracle messages produced by the indexer algorithm for the polynomial IOP using the polynomial commitment scheme;
- the prover algorithm runs the polynomial IOP prover, committing to each oracle message using the polynomial commitment scheme, answering polynomial evaluation queries directly and running the PC-Eval prover algorithm for each evaluation query;
- the verifier algorithm runs the polynomial IOP verifier, and also checks each polynomial evaluation query using the PC-Eval verifier algorithm.

Finally, by Theorem 8.5 there is a polynomial commitment scheme over $R_\bullet$. The result follows by combining the efficiency parameters of the polynomial IOP from Theorem 7.2 with those of the polynomial commitment scheme in Theorem 8.5.

A Properties of sumcheck-friendly commitments

Definition A.1. *A sumcheck-friendly commitment scheme* CM *is* $(K, B_{\mathsf{INV}}, D, \xi)$**-invertible** *if there exists a polynomial-time inverter algorithm* $\mathcal{INV}$ *such that for every security parameter* $\lambda \in \mathbb{N}$*, message length* $n \in \mathbb{N}$*, and polynomial-time algorithm* $\mathcal{A}$*, the following experiment outputs* 1 *with probability* $1 - \mathrm{negl}(\lambda)$.

1. *Sample* $\mathsf{pp}_{\mathsf{CM}} \leftarrow \mathsf{CM.Setup}(1^\lambda, n)$ *and* $\mathsf{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp}_{\mathsf{CM}})$.
2. $\mathcal{A}(\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck})$ *outputs*
   - *an index* $i \in [\ell]$*;*
   - *a challenge vector* $(r_1, \ldots, r_{i-1}) \in \mathcal{C}^{i-1}$*;*
   - *distinct challenges* $r_i^{(1)}, \ldots, r_i^{(K)} \in \mathcal{C}$*;*
   - *polynomials* $p_1, \ldots, p_K \in \mathbb{M}[X_{i+1}, \ldots, X_\ell]$*;*
   - *a polynomial* $Q(X)$ *in* $\mathbb{C}[X]$ *of degree at most* $d^\star_{\mathsf{ck}}$*; and*
   - *a slackness* $c \in \mathbb{S}_{\mathsf{ck}}$.
3. *The experiment outputs* 1 *if and only if one of the following conditions hold:*
   - *there exists* $j \in [K]$ *such that* $D \cdot \|p_j\|_{\mathbb{M}} > B_{\mathsf{INV}}$*;*
   - *there exists* $j \in [K]$ *such that* p_j *is not* ck*-admissible for* $(r_1, \ldots, r_{i-1}, r_i^{(j)})$*;*
   - *there exists* $j \in [K]$ *such that*

$$\phi_{\mathrm{sc}}\left(Q(r_i^{(j)}), \sum_{\omega_{i+1},\ldots,\omega_\ell \in \mathcal{H}} f_{\mathsf{CM}}\Big(p_j(\omega_{i+1}, \ldots, \omega_\ell), P_{\mathsf{ck}}(r_1, \ldots, r_{i-1}, r_i^{(j)}, \omega_{i+1}, \ldots, \omega_\ell), c\Big), c\right) = 1 \ ;$$

   - $\mathcal{INV}(\mathsf{pp}, \mathsf{ck}, \mathcal{A}(\mathsf{pp}_{\mathsf{CM}}, \mathsf{ck}))$ *outputs* ck*-admissible* $p \in \mathbb{M}[X_i, \ldots, X_\ell]$ *with* $\|p\|_{\mathbb{M}} \leq D \cdot \max_{j \in [K]} \|p_j\|_{\mathbb{M}}$ *such that*

$$\phi_{\mathrm{sc}}\left(\sum_{\omega_i \in \mathcal{H}} Q(\omega_i), \sum_{\omega_i,\ldots,\omega_\ell \in \mathcal{H}} f_{\mathsf{CM}}\Big(p(\omega_i, \ldots, \omega_\ell), P_{\mathsf{ck}}(r_1, \ldots, r_{i-1}, \omega_i, \ldots, \omega_\ell), \xi \cdot c\Big), \xi \cdot c\right) = 1 \ .$$

Acknowledgments

We thank Giacomo Fenzi for his detailed comments which helped us to improve the paper. This research was supported in part by: the Berkeley Haas Blockchain Initiative, a donation from the Ethereum Foundation, NSF CISE Expeditions Award CCF-1730628, NSF CAREER 1943347, and gifts from the Alibaba, Amazon Web Services, Ant Group, Astronomer, Ericsson, Facebook, Futurewei, Google, IBM, Intel, Lacework, Microsoft, Nexla, Nvidia, Samsung, Scotiabank, Splunk, and VMware.

References

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. "A Compressed Σ-Protocol Theory for Lattices". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, pp. 549–579.

[ACLMT22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. "Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable". In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO '22. 2022, pp. 102–132.

[AFGHO16] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. "Structure-Preserving Signatures and Commitments to Group Elements". In: *Journal of Cryptology* 29 (2 2016), pp. 363–421.

[AL21] Martin R. Albrecht and Russell W. F. Lai. "Subtractive Sets over Cyclotomic Rings: Limits of Schnorr-like Arguments over Lattices". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, pp. 519–548.

[ALS20] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. "Practical Product Proofs for Lattice Commitments". In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO '20. 2020, pp. 470–499.

[BBBPWM18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P '18. 2018, pp. 315–334.

[BBCPGL18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. "Sub-linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits". In: *Proceedings of the 38th Annual International Cryptology Conference*. CRYPTO '18. 2018, pp. 669–699.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". In: *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*. ICALP '18. 2018, 14:1–14:17.

[BCCGP16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: *Proceedings of the 35th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '16. 2016, pp. 327–357.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 19–46.

[BCGGRS19] Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. "Linear-Size Constant-Query IOPs for Delegating Computation". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019, pp. 494–521.

[BCIOP13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. “Succinct Non-Interactive Arguments via Linear Interactive Proofs”. In: *Proceedings of the 10th Theory of Cryptography Conference*. TCC ’13. 2013, pp. 315–333.

[BCKLN14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. “Better Zero-Knowledge Proofs for Lattice Encryption and Their Application to Group Signatures”. In: *Proceedings of the 20th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT ’14. 2014, pp. 551–572.

[BCL22] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. “Zero-Knowledge Succinct Arguments with a Linear-Time Prover”. In: *Proceedings of the 42nd Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT ’22. 2022, pp. 275–304.

[BCRSVW19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. “Aurora: Transparent Succinct Arguments for R1CS”. In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’19. Full version available at `https://eprint.iacr.org/2018/828`. 2019, pp. 103–128.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. “Interactive Oracle Proofs”. In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC ’16-B. 2016, pp. 31–60.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. “Sumcheck Arguments and their Applications”. In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO ’21. Extended version at `https://eprint.iacr.org/2021/333.pdf`. 2021, pp. 681–710.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. “Transparent SNARKs from DARK Compilers”. In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’20. 2020, pp. 677–706.

[BHRRS20] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. “Public-Coin Zero-Knowledge Arguments with (almost) Minimal Time and Space Overheads”. In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC ’20. 2020, pp. 168–197.

[BHRRS21] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. “Time- and Space-Efficient Arguments from Groups of Unknown Order”. In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO ’21. 2021, pp. 123–152.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. “Lattice-Based SNARGs and Their Application to More Efficient Obfuscation”. In: *Proceedings of the 36th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’17. 2017, pp. 247–277.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. “Quasi-Optimal SNARGs via Linear Multi-Prover Interactive Proofs”. In: *Proceedings of the 37th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT ’18. 2018, pp. 222–255.

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. “A Non-PCP Approach to Succinct Quantum-Safe Zero-Knowledge”. In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO ’20. 2020, pp. 441–469.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. “Algebraic Techniques for Short(er) Exact Lattice-Based Zero-Knowledge Proofs”. In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO ’19. 2019, pp. 176–202.

[BS22] Ward Beullens and Gregor Seiler. “LaBRADOR: Compact Proofs for R1CS from Module-SIS”. In: (2022).

[CFLMR16] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. “Cryptanalysis of the New CLT Multilinear Map over the Integers”. In: *Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’16. 2016, pp. 509–536.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. "Cryptanalysis of GGH15 Multilinear Maps". In: *Proceedings of the 36th Annual International Cryptology Conference*. CRYPTO '16. 2016, pp. 607–628.

[CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. "Zeroizing Attacks on Indistinguishability Obfuscation over CLT13". In: *Proceedings of the 20th IACR International Conference on Practice and Theory in Public-Key Cryptography*. PKC '17. 2017, pp. 41–58.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. "Practical Multilinear Maps over the Integers". In: *Proceedings of the 33rd Annual Cryptology Conference*. CRYPTO '13. 2013, pp. 476–493.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. "New Multilinear Maps Over the Integers". In: *Proceedings of the 35th Annual Cryptology Conference*. CRYPTO '15. 2015, pp. 267–286.

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. "Post-Quantum Succinct Arguments: Breaking the Quantum Rewinding Barriers". In: *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science*. FOCS '21. 2021.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 769–793.

[Con13] Keith Conrad. *Cyclotomic Extensions*. `https://kconrad.math.uconn.edu/math5211s13/handouts/cyclotomic.pdf`. 2013.

[Cor+15] Jean-Sébastien Coron et al. "Zeroizing Without Low-Level Zeroes: New MMAP Attacks and their Limitations". In: *Proceedings of the 35th Annual Cryptology Conference*. CRYPTO '15. 2015, pp. 247–266.

[Dil] URL: `https://pq-crystals.org/dilithium/index.shtml`.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. "Practical Exact Proofs from Lattices: New Techniques to Exploit Fully-Splitting Rings". In: *Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '20. 2020, pp. 259–288.

[ESSLL19] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. "Short Lattice-Based One-out-of-Many Proofs and Applications to Ring Signatures". In: *Proceedings of the 17th International Conference on Applied Cryptography and Network Security*. ACNS '19. 2019, pp. 67–88.

[ESZ22] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. "MatRiCT$^+$: More Efficient Post-Quantum Private Blockchain Payments". In: *Proceedings of the 43rd IEEE Symposium on Security and Privacy*. SP '22. 2022, pp. 1281–1298.

[Fal] URL: `https://falcon-sign.info/`.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. "Candidate Multilinear Maps from Ideal Lattices". In: *Proceedings of the 32nd Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '13. 2013, pp. 1–17.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. "Graph-Induced Multilinear Maps from Lattices". In: *Proceedings of the 12th Theory of Cryptography Conference*. TCC '15. 2015, pp. 498–527.

[GLSTW21] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad Wahby. *Brakedown: Linear-time and post-quantum SNARKs for R1CS*. Cryptology ePrint Archive, Report 2021/1043. 2021.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. "Lattice-Based zk-SNARKs from Square Span Programs". In: *Proceedings of the 25th ACM Conference on Computer and Communications Security*. CCS '18. 2018, pp. 556–573.

[GN08] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '08. 2008, pp. 31–51.

[GW11] Craig Gentry and Daniel Wichs. "Separating Succinct Non-Interactive Arguments From All Falsifiable Assumptions". In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*. STOC '11. 2011, pp. 99–108.

[Gro11] Jens Groth. "Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments". In: *Proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '11. 2011, pp. 431–448.

[ISW21] Yuval Ishai, Hang Su, and David J. Wu. "Shorter and Faster Post-Quantum Designated-Verifier zkSNARKs from Lattices". In: *Proceedings of the 28th ACM Conference on Computer and Communications Security*. CCS '21. 2021, pp. 212–234.

[Kil92] Joe Kilian. "A note on efficient zero-knowledge proofs and arguments". In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. STOC '92. 1992, pp. 723–732.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. "Algebraic Methods for Interactive Proof Systems". In: *Journal of the ACM* 39.4 (1992), pp. 859–868.

[LMR19] Russell W. F. Lai, Giulio Malavolta, and Viktoria Ronge. "Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography". In: *Proceedings of the 26th ACM Conference on Computer and Communications Security*. CCS '19. 2019, pp. 2057–2074.

[LNS20] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "Practical Lattice-Based Zero-Knowledge Proofs for Integer Relations". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. 2020, pp. 1051–1070.

[LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "SMILE: Set Membership from Ideal Lattices with Applications to Ring Signatures and Confidential Transactions". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO' 21. 2021, pp. 611–640.

[LNS22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General". In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO '22. 2022, pp. 71–101.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. "GGHLite: More Efficient Multilinear Maps from Ideal Lattices". In: *Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '14. 2014, pp. 239–256.

[LSTW21] Jonathan Lee, Srinath Setty, Justin Thaler, and Riad Wahby. *Linear-time zero-knowledge SNARKs for R1CS*. Cryptology ePrint Archive, Report 2021/030. 2021.

[Lee21] Jonathan Lee. "Dory: Efficient, Transparent arguments for Generalised Inner Products and Polynomial Commitments". In: *Proceedings of the 19th Theory of Cryptography Conference*. TCC '21. 2021, pp. 1–34.

[MZ18] Fermi Ma and Mark Zhandry. "The MMap Strikes Back: Obfuscation and New Multilinear Maps Immune to CLT13 Zeroizing Attacks". In: *Proceedings of the 16th Theory of Cryptography Conference*. TCC '18. 2018, pp. 513–543.

[NIS16] NIST. *Post-Quantum Cryptography*. 2016. URL: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`.

[NS22] Ngoc Khanh Nguyen and Gregor Seiler. “Practical Sublinear Proofs for R1CS from Lattices”. In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO ’22. 2022, pp. 133–162.

[Nit19] Anca Nitulescu. “Lattice-Based Zero-Knowledge SNARGs for Arithmetic Circuits”. In: *Proceddings of the 6th International Conference on Cryptology and Information Security in Latin America*. LATINCRYPT’ 19. 2019, pp. 217–236.

[PLS18] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. “Lattice-Based Group Signatures and Zero-Knowledge Proofs of Automorphism Stability”. In: *Proceedings of the 25th Conference on Computer and Communications Security*. CCS ’18. 2018, pp. 574–591.

[PLS19] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. “Short Discrete Log Proofs for FHE and Ring-LWE Ciphertexts”. In: *Proceedings of the 22nd International Conference on Practice and Theory of Public-Key Cryptography*. PKC ’19. 2019, pp. 344–373.

[Pic] URL: https://microsoft.github.io/Picnic/.

[RR22] Noga Ron-Zewi and Ron D. Rothblum. “Proving as Fast as Computing: Succinct Arguments with Constant Prover Overhead”. In: *Proceedings of the 54th Annual ACM Symposium on Theory of Computing*. STOC ’22. 2022, pp. 1353–1363.

[RRR21] Omer Reingold, Guy Rothblum, and Ron Rothblum. “Constant-Round Interactive Proofs for Delegating Computation”. In: *SIAM Journal on Computing* 50.3 (2021). Preliminary version appeared in STOC ’16.

[Sph] URL: https://sphincs.org/.

[Tha22] Justin Thaler. *Proofs, Arguments, and Zero-Knowledge*. Unpublished manuscript. 2022. URL: https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. “Orion: Zero Knowledge Proof with Linear Prover Time”. In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO ’22. 2022, pp. 299–328.

[YAZXYW19] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. “Efficient Lattice-Based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications”. In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO ’19. 2019, pp. 147–175.